Multi-layer artificial Neural Network for Estimating Real-Estate Prices

for

SOFT COMPUTING(ITE1015)

SLOT:F1+TF1

in

B.Tech (Information Technology)

By

NITHISHMA.A(17BIT0184)

GARLAPATI SAITEJA(17BIT0217)

FALL SEM 2019-2020

Under the guidance of

Prof. TAPAN KUMAR DAS

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ABSTRACT:

The estimation of real estate prices, are a useful and realistic approach for buyers and for local and fiscal authorities. It is of utmost importance to evaluate the current status of the market and predict its performance over the short term in order to make appropriate financial decisions. We will use two advanced modelling approaches Multi-Level Models and Artificial Neural Networks to model house prices. This approach is compared with the standard Hedonic Price Model in terms of accuracy in prediction, collecting the location information and their explanatory (interpretation) power. This project presents the development of a multi-layer artificial neural network-based models to support real estate investors and home developers in this critical task. The models utilize historical market performance data sets to train the artificial neural networks in order to predict unforeseen future performances. An application example is analyzed to demonstrate the model capabilities in analyzing and predicting the market performance.

Given a set of values describing a house up for sale, a selling price is to be estimated based on the previous data. Before getting into predicting the sale-price of the house, exploratory data analysis will be performed to find out features having the highest weights in determining the same.

BACKGROUND:

1. Neural Network Based Model for Predicting Housing Market Performance.

Reference: TSINGHUA SCIENCE AND TECHNOLOGY ISSN 1007-0214 52/67 pp325-328 Volume 13, Number S1, October 2008.

The United States real estate market is currently facing its worst hit in two decades due to the slowdown of housing sales. The most affected by this decline are real estate investors and home developers who are currently struggling to break-even financially on their investments. For these investors, it is of utmost importance to evaluate the current status of the market and predict its performance over the short-term in order to make appropriate financial decisions. This paper presents the development of artificial neural network-based models to support real estate investors and home developers in this critical task. The paper describes the decision variables, design methodology, and the implementation of these models. The models utilize historical market performance data sets to train the artificial neural networks in order to predict unforeseen future performances. An application example is analyzed to demonstrate the model capabilities in analyzing and predicting the market performance. The model testing and validation showed that the error in prediction is in the range between –2% and +2%.

2. Forecasting the land price using statistical and neural network software.

Reference: 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

This paper focuses on the modelling and forecasting of land price in Chennai Metropolitan Area (CMA) in the state of Tamil nadu, India using multiple regression and neural network techniques. Thirteen locations spread over CMA are selected at random as study areas. The monthly average values of the selected factors from the year 1997 to 2011 are considered to develop the models. Both multiple regression and neural network models are validated with the market price in the year 2012 and 2013. After validation the models are used to forecast the land price in CMA for the years 2014 and 2015. Both the models are found to be well fit for the trend of land price; however, the model using neural network shows better accuracy. A careful examination of the results of forecasting bring to lime light the surge in growth of land prices in the southern and western parts of CMA.

Literature Survey:

Artificial Neural Networks (ANNs) are able to learn, to generalize results and to respond adequately to highly incomplete or previously unknown data (Shaw, 1992). ANN methodology was developed to capture functional forms, allowing the uncovering of hidden non-linear relationships between the variables. This method has been developed in the past years, especially using information of the study area showing outstanding performances. It represents a sub-field of computer science concerned with the use of computers in tasks that are normally considered to require knowledge and cognitive abilities (Gevarter, 1985). It has been applied to the property price forecasting in recent years (Lai Pi-ying, 2011). Borst (1991) has defined a great number of variables in his network to appraise real estate in New York State, demonstrating that ANNs are able to predict the real estate price with 90% accuracy. ANNs perform better than multi-variate analysis, since networks are nonlinear. They can also evaluate subjective information, such as the transport system and the characteristics of the zone, which are difficult to incorporate into traditional mathematical approaches. Traditional multiple regression models have focused on the relationship between real estate prices and accessibility until a systematic review of various research works was carried out by Fujita (1989). Research about how transport system can influence real estate prices was initiated by von Thünen (1826) who laid the foundations of a theory about the distribution of land use and rents in urban areas as proposed by Alonso (1964), Muth (1969) and Mills (1972). Many hedonic studies have specified the role of quality of environment considering the real estate price (Din etal. 2001), accessibility and other local land-use attributes (Ibeas et al., 2012; Chiarazzo et al., 2014). The results highlight a significant influence of variables such as the distance in kilometres to reach the industrial centre or the value of environmental polluters on the variability of the relationship between accessibility to bus stop and real estate prices. The properties located close to the industrial area also showed significant and negative changes in value (Chiarazzo et al., 2014). Other studies have focused on the impact resulted by Bus Rapid Transit systems on real estate prices (Rodríguez and Mojica, 2009). These studies showed the impact on property values resulted by introducing a Bus Rapid Transit (BRT) system in a city and found an increase in price. In this paper, an ANN approach is proposed with an analysis of performances in estimating the sale price of residential properties.

Authors.	Methodology or Technique Used.	Advantages.	Issues.	Metrics.
Ahmed Khalafallah.	The ANN models are designed as feed-forward backpropagation multilayer perception networks using NeuroSolutions	1.robust in approximating almost any input/output. 2.Several network structures are trained, cross-validated and tested by varying the number of hidden layers, the number of neurons in each hidden layer, the transfer function, the learning method, the cross-validation sample size, and the testing sample size.	The main limitation of the developed model is that it is not expected to forecast the behavior of the housing market on the long-term. The future work will include training the ANN models to forecast the performance for periods of 6 and 12 months. However, this will entail utilizing larger sets of data spanning several decades in order to capture the cycles of housing market behavior.	**2114** Total usage since Feb 2013. Citations: 29.

V.Sampathkumara. M.Helen Santhib. J. Vanjinathanc.	NN model is constructed with 13 indicators that are PEs with one bias node as input. All the input values are normalized using the MinMax.	The research focuses on the modelling and forecasting of land price at 13 different locations in CMA with economic and social attributes as influencing factors. The modelling and forecasting of land price in the selected study areas is made using multiple regression and neural network techniques. The data between January 1997 and December 2011 are used in the models.	In PRMSE, both regression and NN models show errors less than 12% which demonstrates the significance of the modelling methods. The low PRMSE values (< 5%) indicate the performance of NN in predicting the system.	Readers:17 Citation Indexes:2
Julia M. Jose M Caridad. Francisco J.	A *MLP,* with one hidden layer2 6:6-6-1:1 was used with the following input	1.Most papers include, as explanatory variables, the house size, its location, age and	Because of these limited number of data and factors in certain narrow range, the	Readers: 622 Citations: 5

	variables: size of the property measured in square meters, age of the building, location index, extras index, community expenses and quality index. They were selected after an identification process with several alternatives. The output layer includes only the transaction price, and there are six neurons in the hidden layer.	availability of garage6, and some additional data. Here, a large set of useful characteristics of each property are considered, summarising some of them in several specialized indices. 2. Sensibility analysis confirms these assertions and, due to redundancy in the variables, a subset of explanatory variables are selected, and are confirmed to be stable over the period analyzed.	model cannot be extended for general applications	
Visit Limsombunchai	The accuracy of this value is determined by	ANNs have the ability to learn and model non-linear and	Firstly, the house price used is not the	Cited by 129

	the total mean square error and then back propagation is used in an attempt to reduce prediction errors, which is done through the adjusting of the connection weights. The performance of the network can be influenced by the number of hidden layers and the number of nodes that are included in each hidden layer.	complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex. ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.	actual sale price but the estimated price due to the difficulty in obtaining the real data from the market. Secondly, this paper considered only the current year’s information of the houses. The time effect of the house price, which could potentially impact the estimated results was ignored (the same house should have different price in different years, assuming that age factor is constant). Finally, the house price could be	

			affected by some other economic factors (such as exchange rate and interest rate) are not included in the estimation.	
Vincenza Chiarazzo Leonardo Caggiania Mario Marinellia Michele Ortonville	In this paper, a model based on Artificial Neural Network (ANN) has been applied to real estate appraisal. Moreover, an evaluation of ANN performances in estimating the sale price of residential properties has been carried out.	An advantage of the proposed approach is that, using ANNs, there is no need to assume explicit functions between input and output of the studies because an ANN learns directly from observed data..	In order to evaluate the most significant input variables, a sensitivity analysis has been carried out. For this purpose, starting from the same dataset, the ANN training phase has been repeated 42 times, eliminating each time one of the 42 input variables.	Cited by 17
Víctor Hugo Masías Mauricio A. Valle	We construct the hedonic models using Random Forest (RF),	Random Forest model produced the best predictions of	First, the sample we used contained only	Referred by 1253

	Support Vector Machine (SVM), Neural Network (NN) and classical multiple Linear Regression (LR) using OLS and compare their predictive performance.	Santiago housing prices. This is consistent with the findings of the international literature, which have demonstrated the superior predictive performance of the RF algorithm for explaining housing prices in other markets.	new housing units and thus did not reflect the whole Santiago housing market. A future study could include housing units of all ages, with unit age then becoming one of the explanatory variables. This would result in a model with better price predictions, more meaningful market segmentation and more accurate measures of variable importance. The second limitation of our research has to do with spatial	

			correlation of the model residuals, particularly in linear models estimated with OLS.	
A. Azadeh B. Ziaei, M. Moghaddam	The Hybrid Fuzzy Linear Regression-Fuzzy Cognitive Map algorithm	This paper presents a hybrid algorithm based on fuzzy linear regression (FLR) and fuzzy cognitive map (FCM) to deal with the problem of forecasting and optimization of housing market fluctuations.	The relations, which relate to the house price, have no values; thus, an influence matrix should be developed in order to determine the relations	Readers: 589 Citations: 4
Hakan Kusan , Osman Aytekin, Ilker Özdemir.	Fuzzy logic inference system	It is seen that of the unit price has a very wide range distribution, while considering the qualitative and statistical properties of the unit price	Because of these limited number of data and factors in certain narrow range, the model cannot be extended for general applications	Readers: 455 Citations:3

		(UP, $/m2), which is the sale price per unit area of the residence as the dependent variable attempted to be explained.		
Dr.Christopher Gan and Dr.Minsoo Lee	hedonic price model	The advantage of the hedonic methods is that they control for the characteristics of properties, thus allowing the analyst to distinguish the impact of changing sample composition from actual property appreciation	model specification procedures, multicollinearity, independent variable interactions, heteroscedasticity, non-linearity and outlier data points can seriously hinder the performance of hedonic price model in real estate valuations.	Readers: 323
Limsombunchai Commerce Division, Lincoln University, Canterbury	The model consists of three main layers: input data layer (example the property	ANNs have the ability to learn and model non-linear and complex relationships, which is really	Artificial neural networks require processors with parallel processing	Readers: 18

8150, New Zealand.	attributes), hidden layer(s) (commonly referred as “black box”), and output layer (estimated house price).	important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.	power, in accordance with their structure. For this reason, the realization of the equipment is dependent. Unexplained behavior of the network: This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network	
MarioMarinelli MicheleOttomanelli	Artificial Neural Network (ANN) has been applied to real estate appraisal.	Artificial neural networks have numerical strength that can perform more than one job at the same time. Artificial neural	ANNs can work with numerical information. Problems have to be translated into numerical values before being	Readers: 332

		networks learn events and make decisions by commenting on similar events.	introduced to ANN	
A.K. Soni 1 , Abdulkadir Abubakar Sadiq2	Neural network is an artificial intelligence model originally designed to replicate the human brain's learning process. The model consists of three main layers: input data layer (example the property attributes), hidden layer(s) (commonly referred as "black box"), and output layer .	Storing information on the entire network : Information such as in traditional programming is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not prevent the network from functioning.	The duration of the network is unknown: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.	Readers: 788
V. Kontrimas and A. Verikas	The ANN (FFBP) Network	A neural network can perform tasks that a linear program can not. When an	The neural network needs training to operate. The architecture of	Readers: 21

		element of the neural network fails, it can continue without any problem by their parallel nature.	a neural network is different from the architecture of microprocessors therefore needs to be emulated.	
Itedal Sabri Hashim Bahia	Artificial Neural Network (ANN) is a neurobiological inspired paradigm that emulates the functioning of the brain based on the way that neurons work, because they are recognized as the cellular elements responsible for the brain information processing	Artificial neural networks have numerical strength that can perform more than one job at the same time. Artificial neural networks learn events and make decisions by commenting on similar events.	The display mechanism to be determined here will directly influence the performance of the network . This depends on the user's ability.	Readers: 45 Citations: 2
] L. Huan and M. Hiroshi,	The ANN (CFBP) Network. CF artificial intelligence model is similar to	They can work fine in case of incomplete information. They do not require knowledge of	Requires high processing time for la Advantages / disadvantages Neural networks have	Cited By: 4

	feedforward backpropagation neural network in using the backpropagation algo rithm for weights updating, but the main symptom of this network is that each layer of neurons related to all previous layer of neurons	the algorithm solving the problem (automatic learning). Process information in a highly parallel way.	a number of advantages. Linear and nonlinear models: Complex linear and nonlinear relationships can be derived using neural networks.	

ALGORITHM USED:

Neural network terminology is inspired by the biological operations of specialized cells called neurons. A neuron is a cell that has several inputs that can be activated by some outside process.

Depending on the amount of activation, the neuron produces its own activity and sends this along its outputs. The artificial equivalent of a neuron is a node (also sometimes called neurons, but I will refer to them as nodes to avoid ambiguity) that receives a set of weighted inputs, processes their sum with its activation function and passes the result of the activation function to nodes further down the graph. Note that it is simpler to represent the input to our activation function as a dot product:

$$\varphi(\sum iwiai)=\varphi(wTa)\varphi(\sum iwiai)=\varphi(wTa)$$

we can use a linear activation function:- identity activation function

$$\varphi(wTa)=wTa\varphi(wTa)=wTa$$

The tanh activation function:

$$\varphi(wTa)=\tanh(wTa)\varphi(wTa)=\tanh(wTa)$$

We can then form a network by chaining these nodes together. Usually this is done in layers - one node layer's outputs are connected to the next layer's inputs (we must take care not to introduce cycles in our network, for reasons that will become clear in the section on backpropagation)

Training in this case involves learning the correct edge weights to produce the target output given the input. The network and its trained weights form a function that operates on input data. With the trained network, we can make predictions given any unlabeled test input.

Steps:

Step 0: Load The Data

Step 1: Dataset Summary & Exploration

Step 2: Design and Test a Model Architecture

Step 2.1:Pre-process the Data Set

Step 2.2:Data augmentation

Step 2.3 :Train, Validate and Test the Model

Step 3: Test a Model on by cross validating the dataset with neural network

Step 4 : Visualize the Neural Network's State by fitting the dataset values

Step 0: Load The Data

Step 1: Dataset Summary & Exploration

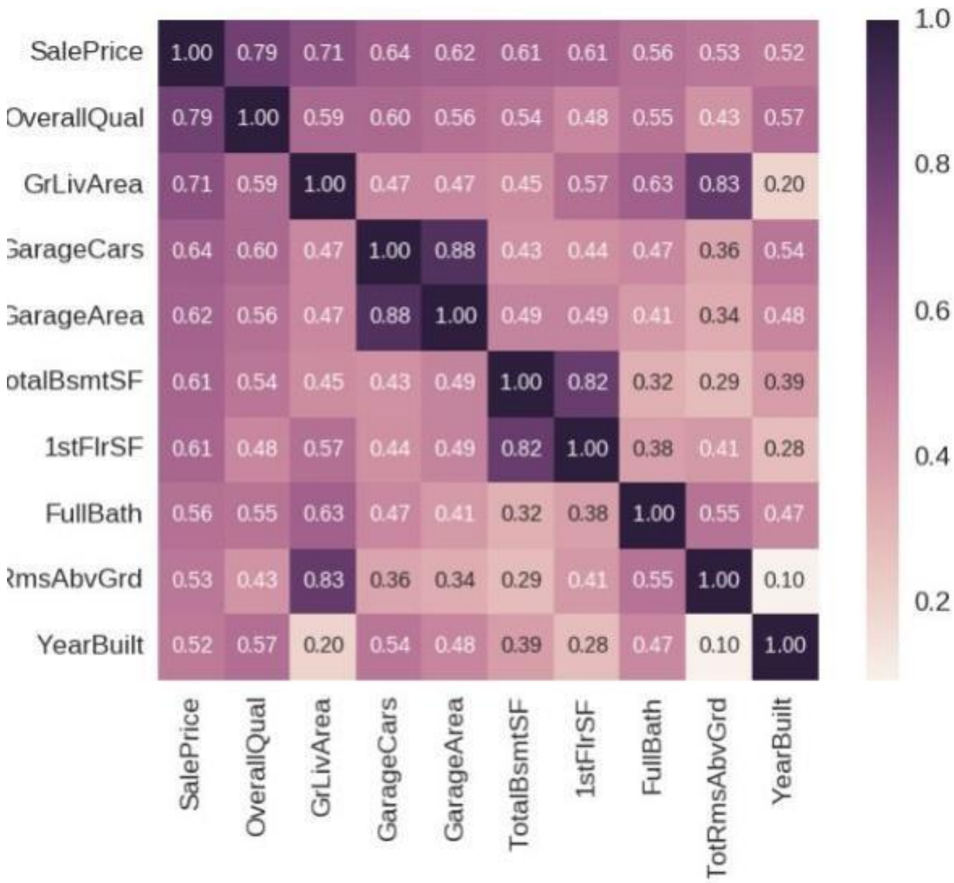
The dataset is provided by Zillow a real estate company based in US. It has price and house description of 1.46K rows. The following image shows the most co-related features with sale-price.

Step 2: Design and Test a Model Architecture

Design and implement a neural network model that learns to estimate the real estate prices. Train and test your model on the Zillow's real estate dataset .

There are various aspects to consider when thinking about this problem:

- Neural network architecture (is the network over or underfitting?)



Play around preprocessing techniques (normalization, rgb to grayscale, etc) Number of examples per label (some have more than others). Generate fake data.

Step 2.1:Pre-process the Data Set

- First finding the missing values in the dataset and dropping those values
- Shortlisting the columns which are much correlated with the sale price

Step 2.2:Data augmentation

The first thing I tried is to augment the data by removing the nnull values by dropping the columns in the dataset and taking the columns which are more correlated with the label value and having less null values

Step 2.3 :Train, Validate and Test the Model

A validation set can be used to assess how well the model is performing. A low accuracy on the training and validation sets imply underfitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting.

Step 3: Test a Model on by cross validating the dataset with neural network

By constructing a 4 layer neural network and fitting the dataset feature and label value we can find the mean squared error after every epoch

Step 4 : Visualize the Neural Network's State by fitting the dataset values

While neural networks can be a great learning device they are often referred to as a black box. We can understand what the weights of a neural network look like better by plotting their feature maps. After successfully training your neural network you can see what it's feature maps look like by plotting the output of the network's weight layers in response to a test the loss. From these plotted feature maps, it's possible to see what characteristics of an dataset the network finds interesting. For a sign, maybe the inner network feature maps react with high activation to the sign's boundary outline or to the contrast in the sign's painted symbol.

Provided for you below is the function code that allows you to get the visualization output of any tensorflow weight layer you want. The inputs to the function should be a shortlisted dataset, one used during training or a new one you provided, and then the tensorflow variable name that represents the layer's state during the training process.

Briefly:

1. Start
2. Import required libraries i.e., pandas, sklearn, matplotlib etc..,
3. Import the "train" csv file with the help of pandas as df_train
4. Take required columns from the train csv file which shows major change with the label value
5. Then shortlisted columns are taken with the highest correlation with the label value(Sale value)
6. Detect the null values and drop those values
7. Selecting important features which makes big contribution to label as imp_feats
8. Train a neural network with first layer of 256 and second layer of 64 inputs and third layer of 32 inputs and the final layer of single input
9. Fit the values of shortlisted X and the label Y with 5000 epochs as history
10. By doing history.history we get the values of 'val_loss', 'val_mean_absolute_percentage_error', 'loss', 'mean_absolute_percentage_error' of each epoch
11. We get the model accuracy with plotting val_mean_absolute_percentage_error and 'mean_absolute_percentage_error with xlabel as epoch and ylabel as mean percentage error

    We get the model loss with plotting loss and val_loss with xlabel as epoch and ylabel as loss .

SOFTWARE USED:

Jupyter software will be used for implementing the code as it is a web application used to create live codes for machine learning. We would develop the multi-layer artificial neural network for real-estate price estimation using python language.

EXPECTED RESULTS:

The expected outcome of our multi-layer artificial neural network is to create a model that would predict the prices of the houses up for sale. After the completion of our project, we would expect our model to estimate the real-estate prices based on the training data with least mean absolute percentage error(MAPE) score.

- ***PCA code***

```
import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

%matplotlib inline

# Load the train data in a dataframe

train = pd.read_csv(r'C:\Users\Nithishma\Desktop\train.csv')

test = pd.read_csv(r'C:\Users\Nithishma\Desktop\test.csv')

train.info()
```

```
nulls = train.isnull().sum().sort_values(ascending=False)

nulls.head(20)

train = train.drop(['Id','PoolQC','MiscFeature','Alley','Fence'],axis = 1)

train[['Fireplaces','FireplaceQu']].head(10)

train['FireplaceQu'].isnull().sum()

train['Fireplaces'].value_counts()

train['FireplaceQu']=train['FireplaceQu'].fillna('NF')

train['LotFrontage']
=train['LotFrontage'].fillna(value=train['LotFrontage'].mean())

train['GarageType'].isnull().sum()

train['GarageCond'].isnull().sum()

train['GarageFinish'].isnull().sum()

train['GarageYrBlt'].isnull().sum()

train['GarageQual'].isnull().sum()

train['GarageArea'].value_counts().head()

train['GarageType']=train['GarageType'].fillna('NG')

train['GarageCond']=train['GarageCond'].fillna('NG')

train['GarageFinish']=train['GarageFinish'].fillna('NG')

train['GarageYrBlt']=train['GarageYrBlt'].fillna('NG')

train['GarageQual']=train['GarageQual'].fillna('NG')

train.BsmtExposure.isnull().sum()

train.BsmtFinType2.isnull().sum()
```

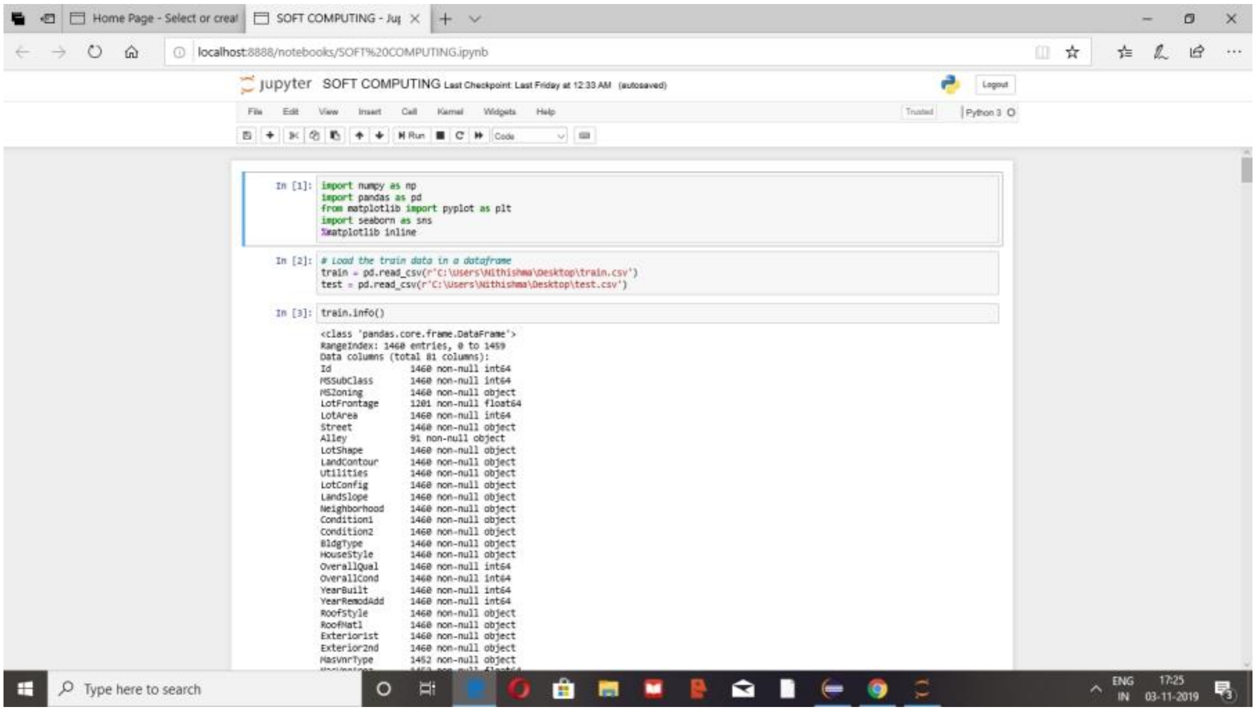
```
train.BsmtFinType1.isnull().sum()

train.BsmtCond.isnull().sum()

train.BsmtQual.isnull().sum()

train.TotalBsmtSF.value_counts().head()

train.TotalBsmtSF.value_counts().head()

train['BsmtExposure']=train['BsmtExposure'].fillna('NB')

train['BsmtFinType2']=train['BsmtFinType2'].fillna('NB')

train['BsmtFinType1']=train['BsmtFinType1'].fillna('NB')

train['BsmtCond']=train['BsmtCond'].fillna('NB')

train['BsmtQual']=train['BsmtQual'].fillna('NB')

train['MasVnrArea'] = train['MasVnrArea'].fillna(train['MasVnrArea'].mean())

train['MasVnrType'] = train['MasVnrType'].fillna('none')

train.Electrical = train.Electrical.fillna('SBrkr')

train.isnull().sum().sum()

num_train = train._get_numeric_data()

num_train.columns

def var_summary(x):

    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),
x.std(), x.var(), x.min(), x.quantile(0.01),
x.quantile(0.05),x.quantile(0.10),x.quantile(0.25),x.quantile(0.50),x.quantile(0.
75), x.quantile(0.90),x.quantile(0.95), x.quantile(0.99),x.max()],

index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1' , 'P5'
,'P10' ,'P25' ,'P50' ,'P75' ,'P90' ,'P95' ,'P99' ,'MAX'])
```

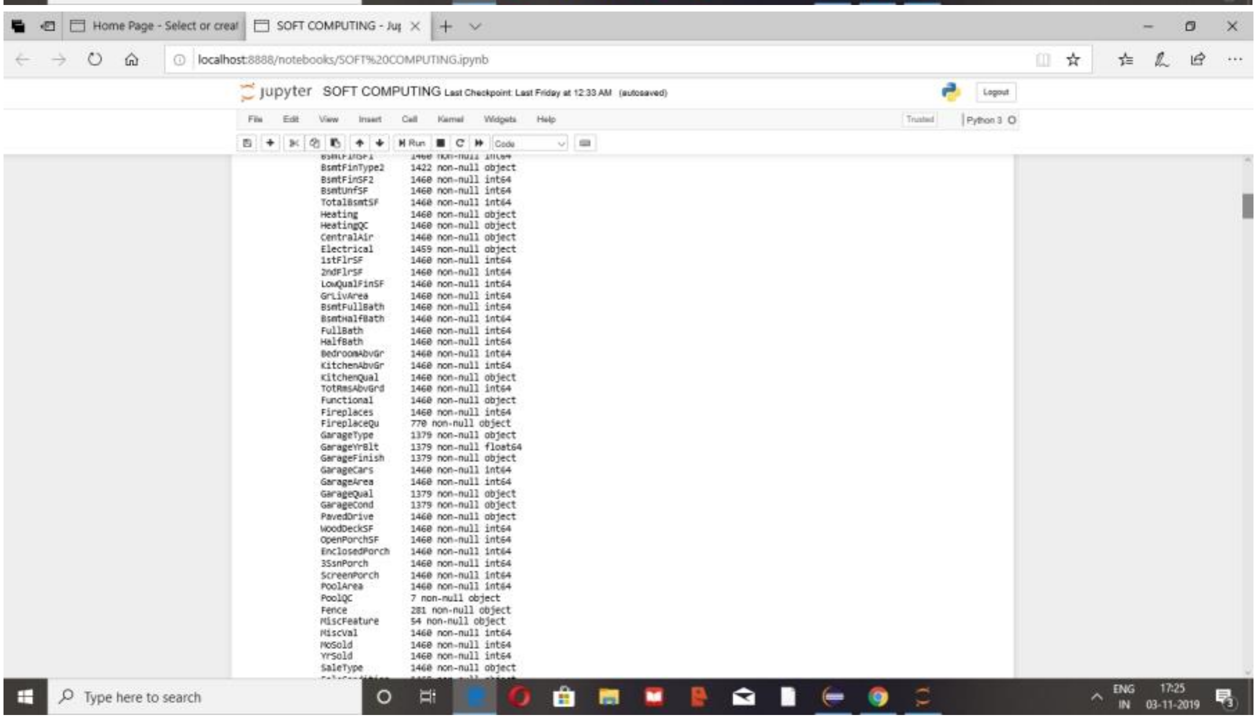
```
num_train.apply(lambda x: var_summary(x)).T

sns.boxplot([num_train.LotFrontage])

train['LotFrontage']=
train['LotFrontage'].clip(upper=train['LotFrontage'].quantile(0.99))

sns.boxplot(num_train.LotArea)

train['LotArea']= train['LotArea'].clip(upper=train['LotArea'].quantile(0.99))

sns.boxplot(train['MasVnrArea'])

train['MasVnrArea']=
train['MasVnrArea'].clip(upper=train['MasVnrArea'].quantile(0.99))

sns.boxplot(train['BsmtFinSF1'])

sns.boxplot(train['BsmtFinSF2'])

train['BsmtFinSF1']=
train['BsmtFinSF1'].clip(upper=train['BsmtFinSF1'].quantile(0.99))

train['BsmtFinSF2']=
train['BsmtFinSF2'].clip(upper=train['BsmtFinSF2'].quantile(0.99))

sns.boxplot(train['TotalBsmtSF'])

train['TotalBsmtSF']=
train['TotalBsmtSF'].clip(upper=train['TotalBsmtSF'].quantile(0.99))

sns.boxplot(train['1stFlrSF'])

train['1stFlrSF']= train['1stFlrSF'].clip(upper=train['1stFlrSF'].quantile(0.99))

sns.boxplot(train['2ndFlrSF'])

train['2ndFlrSF']= train['2ndFlrSF'].clip(upper=train['2ndFlrSF'].quantile(0.99))

sns.boxplot(train['GrLivArea'])
```

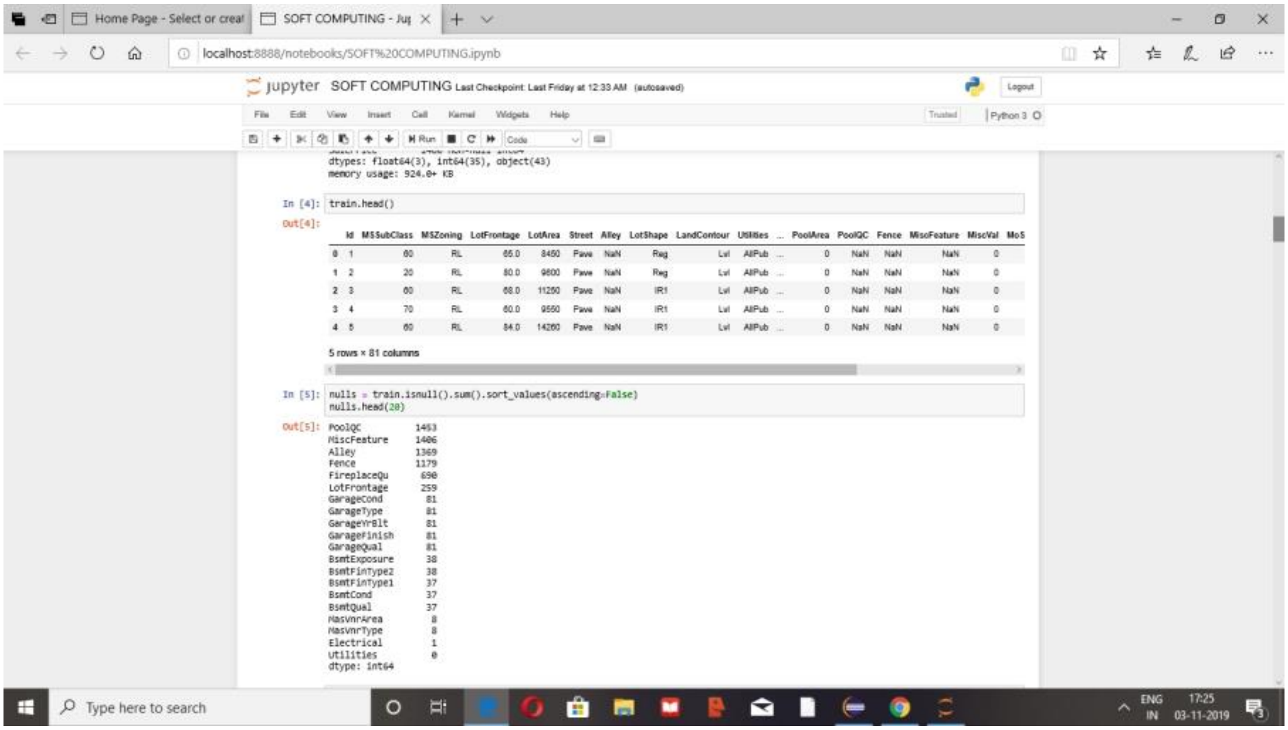
```
train['GrLivArea']=
train['GrLivArea'].clip(upper=train['GrLivArea'].quantile(0.99))

sns.boxplot(train['BedroomAbvGr'])

train['BedroomAbvGr']=
train['BedroomAbvGr'].clip(upper=train['BedroomAbvGr'].quantile(0.99))

train['BedroomAbvGr']=
train['BedroomAbvGr'].clip(lower=train['BedroomAbvGr'].quantile(0.01))

sns.boxplot(train['GarageCars'])

train['GarageCars']=
train['GarageCars'].clip(upper=train['GarageCars'].quantile(0.99))

sns.boxplot(train['GarageArea'])

train['GarageArea']=
train['GarageArea'].clip(upper=train['GarageArea'].quantile(0.99))

sns.boxplot(train['WoodDeckSF'])

train['WoodDeckSF']=
train['WoodDeckSF'].clip(upper=train['WoodDeckSF'].quantile(0.99))

sns.boxplot(train['OpenPorchSF'])

train['OpenPorchSF']=
train['OpenPorchSF'].clip(upper=train['OpenPorchSF'].quantile(0.99))

sns.boxplot(train['EnclosedPorch'])

train['EnclosedPorch']=
train['EnclosedPorch'].clip(upper=train['EnclosedPorch'].quantile(0.99))

sns.boxplot(train['3SsnPorch'])

train['3SsnPorch']=
train['3SsnPorch'].clip(upper=train['3SsnPorch'].quantile(0.99))
```

```
sns.boxplot(train['ScreenPorch'])

train['ScreenPorch']=
train['ScreenPorch'].clip(upper=train['ScreenPorch'].quantile(0.99))

sns.boxplot(train['PoolArea'])

train['PoolArea']= train['PoolArea'].clip(upper=train['PoolArea'].quantile(0.99))

sns.boxplot(train['MiscVal'])

sns.boxplot(train.SalePrice)

train['SalePrice']= train['SalePrice'].clip(upper=train['SalePrice'].quantile(0.99))

train['SalePrice']= train['SalePrice'].clip(lower=train['SalePrice'].quantile(0.01))

train['MiscVal']= train['MiscVal'].clip(upper=train['MiscVal'].quantile(0.99))

num_corr=num_train .corr()

plt.subplots(figsize=(13,10))

sns.heatmap(num_corr,vmax =.8 ,square = True)

k = 14

cols = num_corr.nlargest(k, 'SalePrice')['SalePrice'].index

cm = np.corrcoef(num_train[cols].values.T)

sns.set(font_scale=1.35)

f, ax = plt.subplots(figsize=(10,10))

hm=sns.heatmap(cm, annot = True,vmax =.8, yticklabels=cols.values,
xticklabels = cols.values)

from sklearn.preprocessing import StandardScaler

train_d = pd.get_dummies(train)
```

```
train_d1 = train_d.drop(['SalePrice'],axis = 1)

y = train_d.SalePrice

scaler = StandardScaler()

scaler.fit(train_d1)

t_train = scaler.transform(train_d1)

from sklearn.decomposition import PCA

pca_hp = PCA(30)

x_fit = pca_hp.fit_transform(t_train)

np.exp(pca_hp.explained_variance_ratio_)
```


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
In [2]: # Load the train data in a dataframe
train = pd.read_csv(r'C:\Users\Nithishma\Desktop\train.csv')
test = pd.read_csv(r'C:\Users\Nithishma\Desktop\test.csv')
In [3]: train.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id 1460 non-null int64
MSSubClass 1460 non-null int64
MSZoning 1460 non-null object
LotFrontage 1201 non-null float64
LotArea 1460 non-null int64
Street 1460 non-null object
Alley 91 non-null object
LotShape 1460 non-null object
LandContour 1460 non-null object
Utilities 1460 non-null object
LotConfig 1460 non-null object
LandSlope 1460 non-null object
Neighborhood 1460 non-null object
Condition1 1460 non-null object
Condition2 1460 non-null object
BldgType 1460 non-null object
HouseStyle 1460 non-null object
OverallQual 1460 non-null int64
OverallCond 1460 non-null int64
YearBuilt 1460 non-null int64
YearRemodAdd 1460 non-null int64
RoofStyle 1460 non-null object
RoofMatl 1460 non-null object
Exterior1st 1460 non-null object
Exterior2nd 1460 non-null object
MasVnrType 1452 non-null object
Type here to search
ENG
IN
17:25
03-11-2019


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
BsmtFinType2 1422 non-null object
BsmtFinSF2 1460 non-null int64
BsmtUnfSF 1460 non-null int64
TotalBsmtSF 1460 non-null int64
Heating 1460 non-null object
HeatingQC 1460 non-null object
CentralAir 1460 non-null object
Electrical 1459 non-null object
1stFlrSF 1460 non-null int64
2ndFlrSF 1460 non-null int64
LowQualFinSF 1460 non-null int64
GrLivArea 1460 non-null int64
BsmtFullBath 1460 non-null int64
BsmtHalfBath 1460 non-null int64
FullBath 1460 non-null int64
HalfBath 1460 non-null int64
BedroomAbvGr 1460 non-null int64
KitchenAbvGr 1460 non-null int64
KitchenQual 1460 non-null object
TotRmsAbvGrd 1460 non-null int64
Functional 1460 non-null object
Fireplaces 1460 non-null int64
FireplaceQu 770 non-null object
GarageType 1379 non-null object
GarageYrBlt 1379 non-null float64
GarageFinish 1379 non-null object
GarageCars 1460 non-null int64
GarageArea 1460 non-null int64
GarageQual 1379 non-null object
GarageCond 1379 non-null object
PavedDrive 1460 non-null object
WoodDeckSF 1460 non-null int64
OpenPorchSF 1460 non-null int64
EnclosedPorch 1460 non-null int64
3SsnPorch 1460 non-null int64
ScreenPorch 1460 non-null int64
PoolArea 1460 non-null int64
PoolQC 7 non-null object
Fence 281 non-null object
MiscFeature 54 non-null object
MiscVal 1460 non-null int64
MoSold 1460 non-null int64
YrSold 1460 non-null int64
SaleType 1460 non-null object
Type here to search
ENG
IN
17:25
03-11-2019



jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)

```
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [4]:
```
train.head()
```

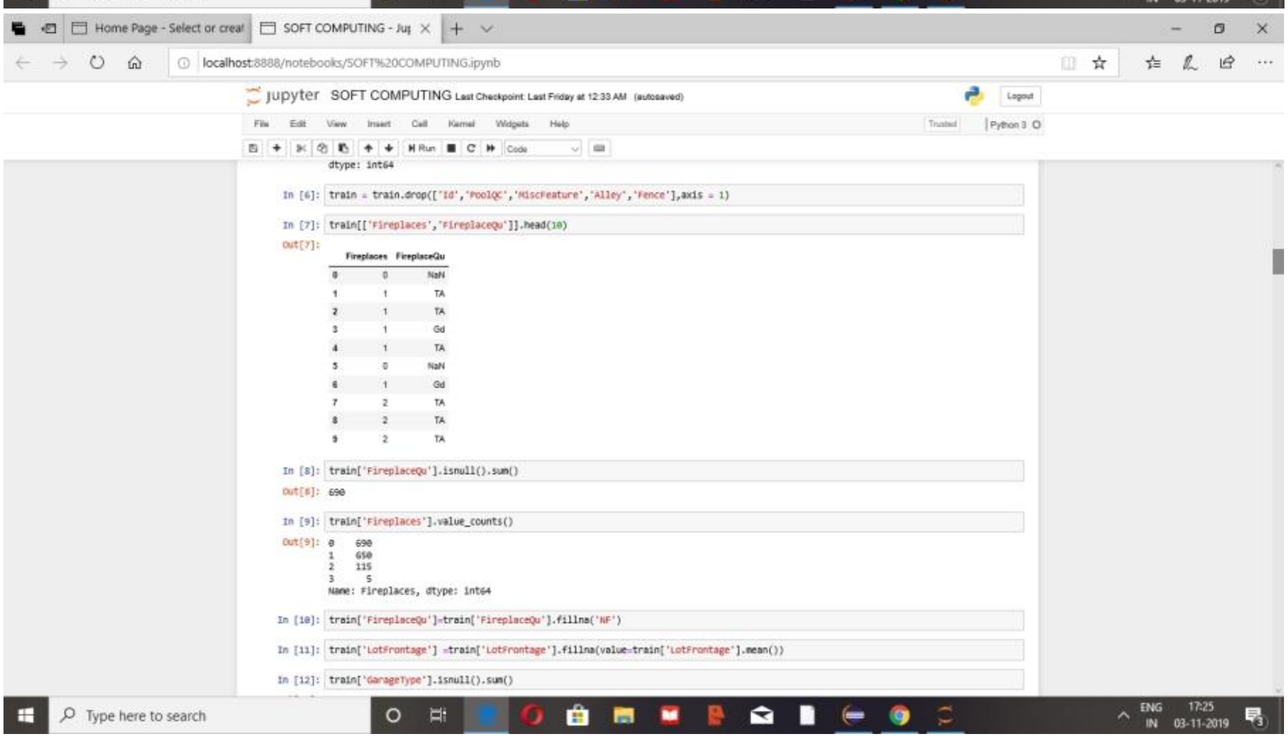
Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoS
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows × 81 columns

In [5]:
```
nulls = train.isnull().sum().sort_values(ascending=False)
nulls.head(20)
```

Out[5]:
```
PoolQC          1453
MiscFeature     1406
Alley           1369
Fence           1179
FireplaceQu      690
LotFrontage      259
GarageCond        81
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
BsmtExposure      38
BsmtFinType2      38
BsmtFinType1      37
BsmtCond          37
BsmtQual          37
MasVnrArea         8
MasVnrType         8
Electrical         1
Utilities          0
dtype: int64
```



jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)

```
dtype: int64
```

In [6]:
```
train = train.drop(['Id','PoolQC','MiscFeature','Alley','Fence'],axis = 1)
```

In [7]:
```
train[['Fireplaces','FireplaceQu']].head(10)
```

Out[7]:

	Fireplaces	FireplaceQu
0	0	NaN
1	1	TA
2	1	TA
3	1	Gd
4	1	TA
5	0	NaN
6	1	Gd
7	2	TA
8	2	TA
9	2	TA

In [8]:
```
train['FireplaceQu'].isnull().sum()
```

Out[8]: 690

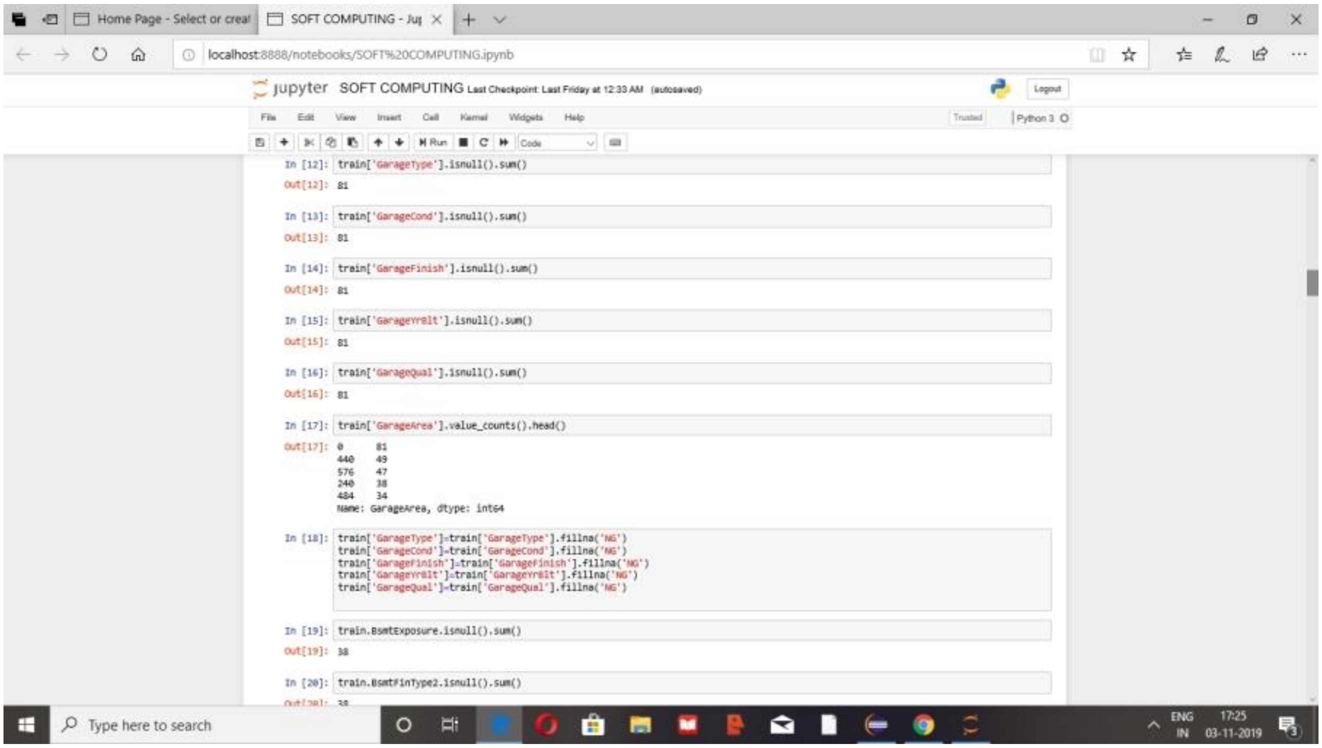
In [9]:
```
train['Fireplaces'].value_counts()
```

Out[9]:
```
0    690
1    650
2    115
3      5
Name: Fireplaces, dtype: int64
```

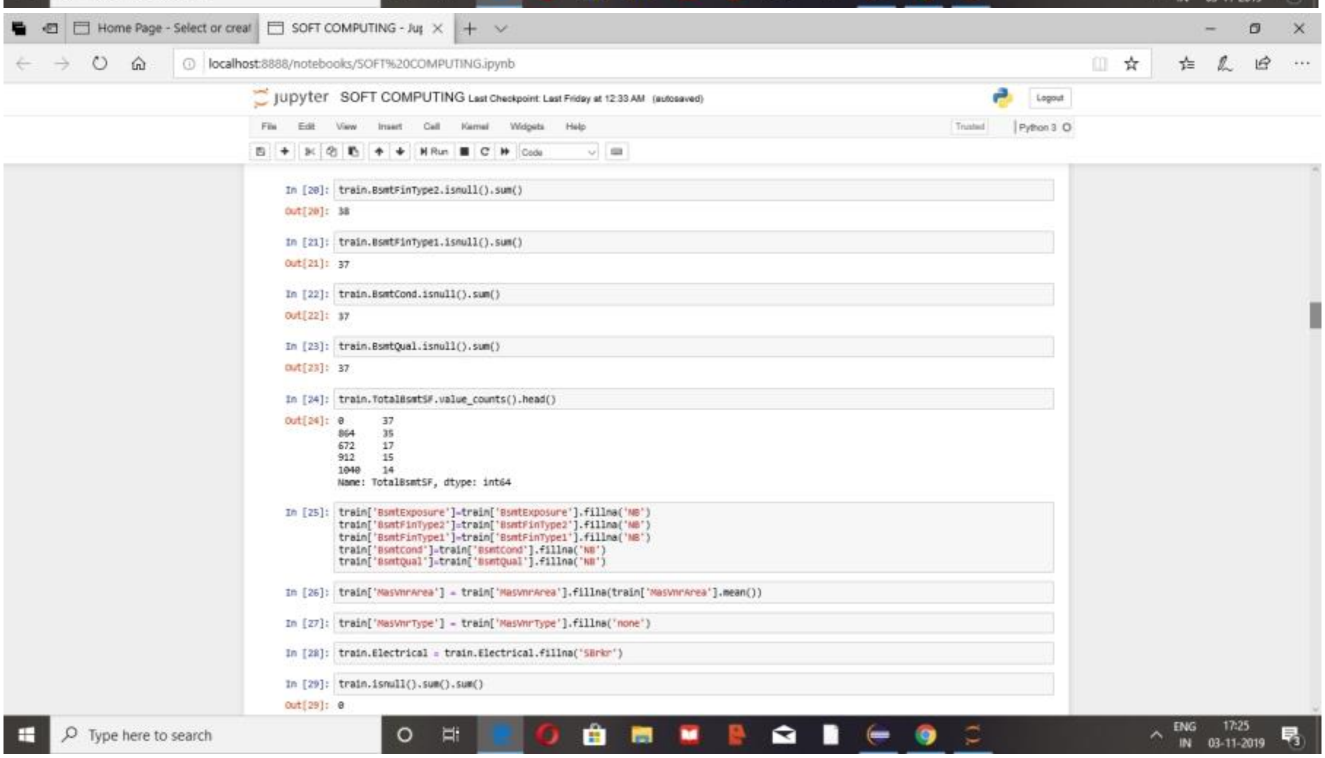
In [10]:
```
train['FireplaceQu']=train['FireplaceQu'].fillna('NF')
```

In [11]:
```
train['LotFrontage'] =train['LotFrontage'].fillna(value=train['LotFrontage'].mean())
```

In [12]:
```
train['GarageType'].isnull().sum()
```



```
In [12]: train['GarageType'].isnull().sum()
Out[12]: 81

In [13]: train['GarageCond'].isnull().sum()
Out[13]: 81

In [14]: train['GarageFinish'].isnull().sum()
Out[14]: 81

In [15]: train['GarageYrBlt'].isnull().sum()
Out[15]: 81

In [16]: train['GarageQual'].isnull().sum()
Out[16]: 81

In [17]: train['GarageArea'].value_counts().head()
Out[17]: 0      81
         440    49
         576    47
         240    38
         484    34
         Name: GarageArea, dtype: int64

In [18]: train['GarageType']=train['GarageType'].fillna('NG')
         train['GarageCond']=train['GarageCond'].fillna('NG')
         train['GarageFinish']=train['GarageFinish'].fillna('NG')
         train['GarageYrBlt']=train['GarageYrBlt'].fillna('NG')
         train['GarageQual']=train['GarageQual'].fillna('NG')

In [19]: train.BsmtExposure.isnull().sum()
Out[19]: 38

In [20]: train.BsmtFinType2.isnull().sum()
Out[20]: 38
```



```
In [20]: train.BsmtFinType2.isnull().sum()
Out[20]: 38

In [21]: train.BsmtFinType1.isnull().sum()
Out[21]: 37

In [22]: train.BsmtCond.isnull().sum()
Out[22]: 37

In [23]: train.BsmtQual.isnull().sum()
Out[23]: 37

In [24]: train.TotalBsmtSF.value_counts().head()
Out[24]: 0       37
         864     35
         672     17
         912     15
         1040    14
         Name: TotalBsmtSF, dtype: int64

In [25]: train['BsmtExposure']=train['BsmtExposure'].fillna('NB')
         train['BsmtFinType2']=train['BsmtFinType2'].fillna('NB')
         train['BsmtFinType1']=train['BsmtFinType1'].fillna('NB')
         train['BsmtCond']=train['BsmtCond'].fillna('NB')
         train['BsmtQual']=train['BsmtQual'].fillna('NB')

In [26]: train['MasVnrArea'] = train['MasVnrArea'].fillna(train['MasVnrArea'].mean())

In [27]: train['MasVnrType'] = train['MasVnrType'].fillna('none')

In [28]: train.Electrical = train.Electrical.fillna('SBrkr')

In [29]: train.isnull().sum().sum()
Out[29]: 0
```

localhost:8888/notebooks/SOFT%20COMPUTING.ipynb

jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)

In [30]:
```
num_train = train._get_numeric_data()
```

In [31]:
```
num_train.columns
```

Out[31]:
```
Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
       'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
       'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
       'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [32]:
```
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),  x.std(), x.var(), x.min(), x.quantile(0.01), 
                  index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1' , 'P5' ,'P10' ,'P25' ,'P50' ,'P75' ,'P9

num_train.apply(lambda x: var_summary(x)).T
```

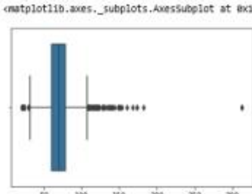
Out[32]:

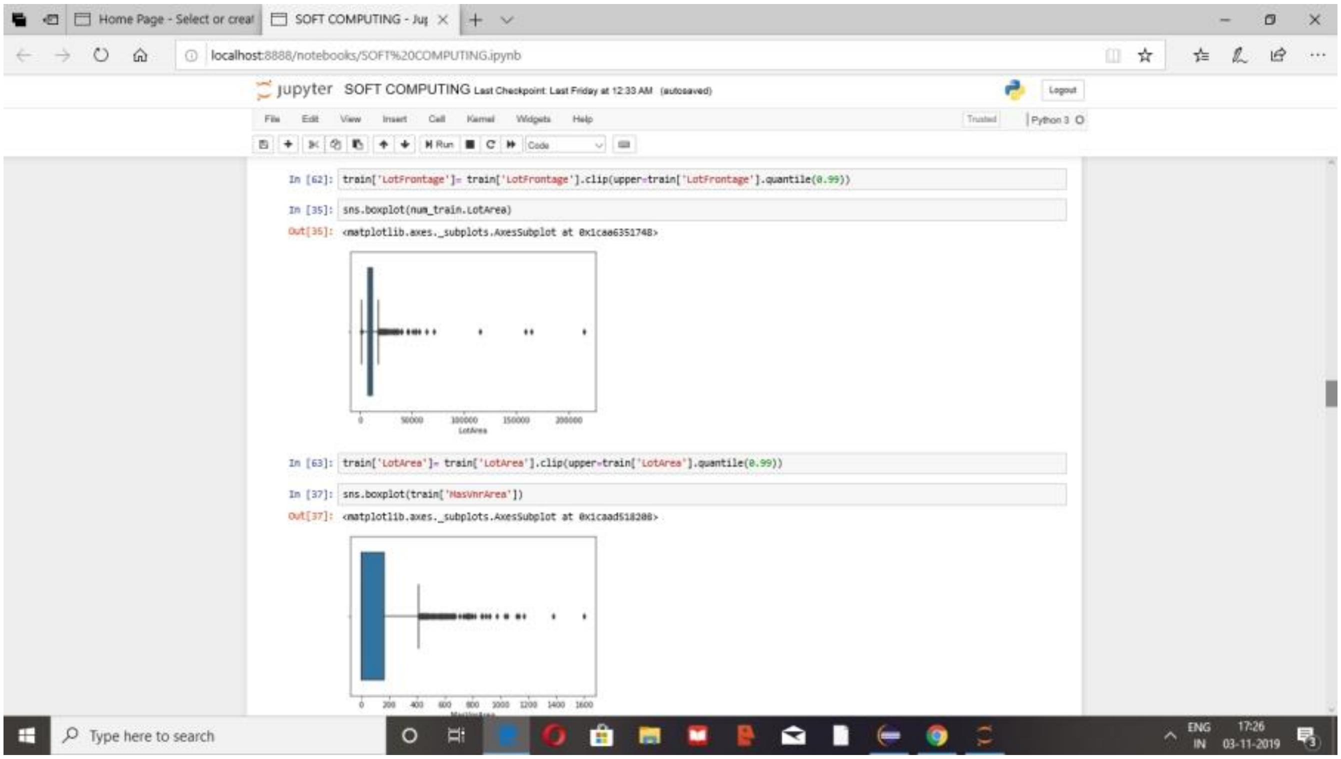
	N	NMISS	SUM	MEAN	MEDIAN	STD	VAR	MIN	P1	P5	P10	P25
MSSubClass	1460.0	0.0	8.307000e+04	56.897260	50.000000	42.300571	1.789338e+03	20.0	20.00	20.00	20.0	20.00
LotFrontage	1460.0	0.0	1.022729e+05	70.049958	70.049958	22.024023	4.850576e+02	21.0	21.00	35.95	49.0	60.00
LotArea	1460.0	0.0	1.535457e+07	10516.828082	9478.500000	9981.264932	9.962565e+07	1300.0	1680.00	3311.70	5000.0	7553.50
OverallQual	1460.0	0.0	8.905000e+03	6.099315	6.000000	1.382997	1.912679e+00	1.0	3.00	4.00	5.0	5.00
OverallCond	1460.0	0.0	8.140000e+03	5.575342	5.000000	1.112799	1.238322e+00	1.0	3.00	4.00	5.0	5.00
YearBuilt	1460.0	0.0	2.878051e+06	1971.267808	1973.000000	30.202904	9.122154e+02	1872.0	1899.18	1916.00	1924.9	1954.00
YearRemodAdd	1460.0	0.0	2.897904e+06	1984.865753	1994.000000	20.645407	4.262328e+02	1950.0	1950.00	1950.00	1950.0	1967.00
MasVnrArea	1460.0	0.0	1.513805e+05	103.685262	0.000000	180.569112	3.260520e+04	0.0	0.00	0.00	0.0	0.00
BsmtFinSF1	1460.0	0.0	6.477140e+05	443.639726	383.500000	456.098091	2.080255e+05	0.0	0.00	0.00	0.0	0.00
BsmtFinSF2	1460.0	0.0	6.798200e+04	46.549315	0.000000	161.319273	2.602391e+04	0.0	0.00	0.00	0.0	0.00
BsmtUnfSF	1460.0	0.0	8.281710e+05	567.240411	477.500000	441.866955	1.952464e+05	0.0	0.00	0.00	74.9	223.00
TotalBsmtSF	1460.0	0.0	1.543847e+06	1057.429452	991.500000	438.705324	1.924624e+05	0.0	0.00	519.30	636.9	795.75
1stFlrSF	1460.0	0.0	1.697435e+06	1162.626712	1087.000000	386.587738	1.494501e+05	334.0	520.00	672.95	756.9	882.00
2ndFlrSF	1460.0	0.0	5.066090e+05	346.992466	0.000000	436.528436	1.905571e+05	0.0	0.00	0.00	0.0	0.00

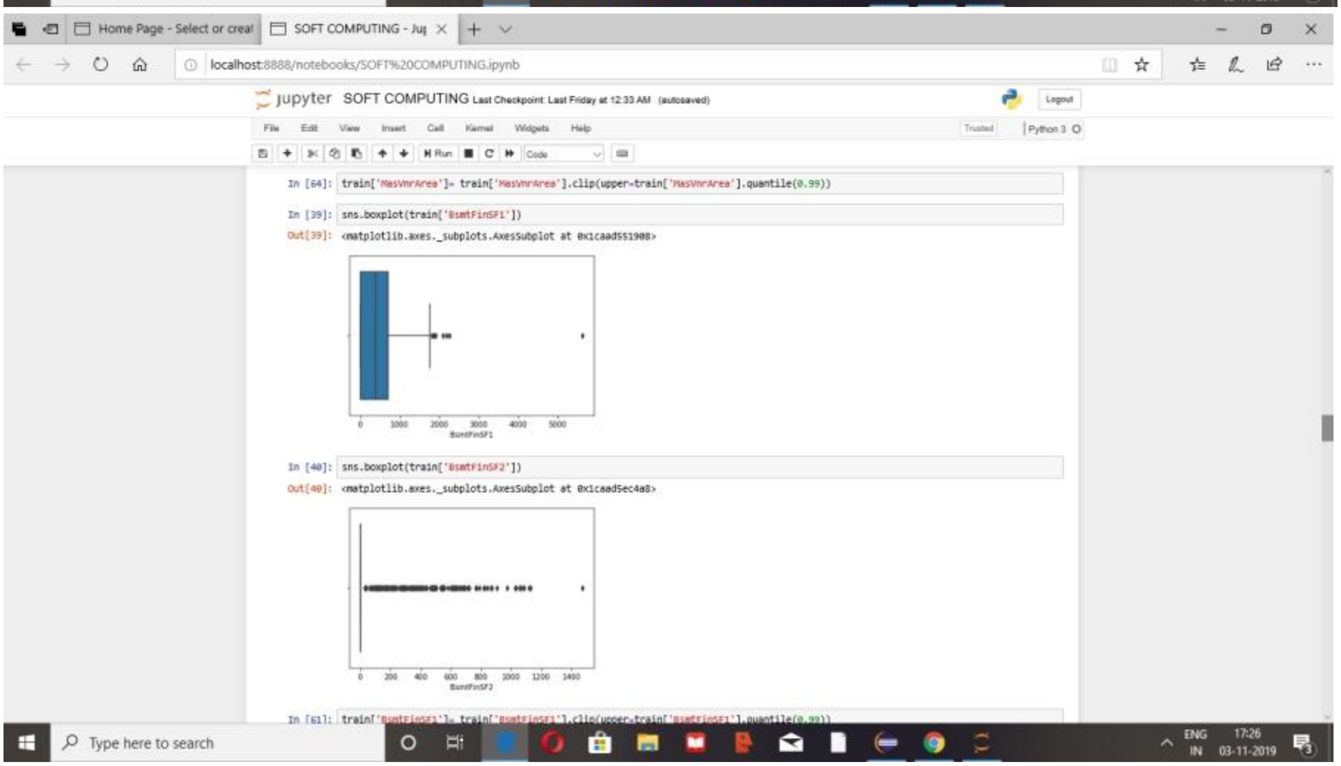
BedroomAbvGr	1460.0	0.0	[illegible]	2.866438	3.000000	0.815778	6.654938e-01	0.0	1.00	2.00	2.0	2.00
KitchenAbvGr	1460.0	0.0	1.528000e+03	1.046575	1.000000	0.220338	4.854892e-02	0.0	1.00	1.00	1.0	1.00
TotRmsAbvGrd	1460.0	0.0	9.516000e+03	6.517808	6.000000	1.625393	2.641903e+00	2.0	3.00	4.00	5.0	5.00
Fireplaces	1460.0	0.0	8.950000e+02	0.613014	1.000000	0.644666	4.155947e-01	0.0	0.00	0.00	0.0	0.00
GarageCars	1460.0	0.0	2.580000e+03	1.767123	2.000000	0.747315	5.584797e-01	0.0	0.00	0.00	1.0	1.00
GarageArea	1460.0	0.0	6.905510e+05	472.980137	480.000000	213.804841	4.571251e+04	0.0	0.00	0.00	240.0	334.50
WoodDeckSF	1460.0	0.0	1.375970e+05	94.244521	0.000000	125.338794	1.570981e+04	0.0	0.00	0.00	0.0	0.00
OpenPorchSF	1460.0	0.0	6.812400e+04	46.660274	25.000000	66.256028	4.389861e+03	0.0	0.00	0.00	0.0	0.00
EnclosedPorch	1460.0	0.0	3.205300e+04	21.954110	0.000000	61.119149	3.735550e+03	0.0	0.00	0.00	0.0	0.00
3SsnPorch	1460.0	0.0	4.978000e+03	3.409589	0.000000	29.317331	8.595059e+02	0.0	0.00	0.00	0.0	0.00
ScreenPorch	1460.0	0.0	2.198900e+04	15.060959	0.000000	55.757415	3.108889e+03	0.0	0.00	0.00	0.0	0.00
PoolArea	1460.0	0.0	4.028000e+03	2.758904	0.000000	40.177307	1.614216e+03	0.0	0.00	0.00	0.0	0.00
MiscVal	1460.0	0.0	6.349400e+04	43.489041	0.000000	496.123024	2.461381e+05	0.0	0.00	0.00	0.0	0.00
MoSold	1460.0	0.0	9.230000e+03	6.321918	6.000000	2.703626	7.309595e+00	1.0	1.00	2.00	3.0	5.00
YrSold	1460.0	0.0	2.931411e+06	2007.815753	2008.000000	1.328095	1.763837e+00	2006.0	2006.00	2006.00	2006.0	2007.00
SalePrice	1460.0	0.0	2.641449e+08	180921.195890	163000.000000	79442.502883	6.311111e+09	34900.0	61815.97	88000.00	106475.0	129975.00

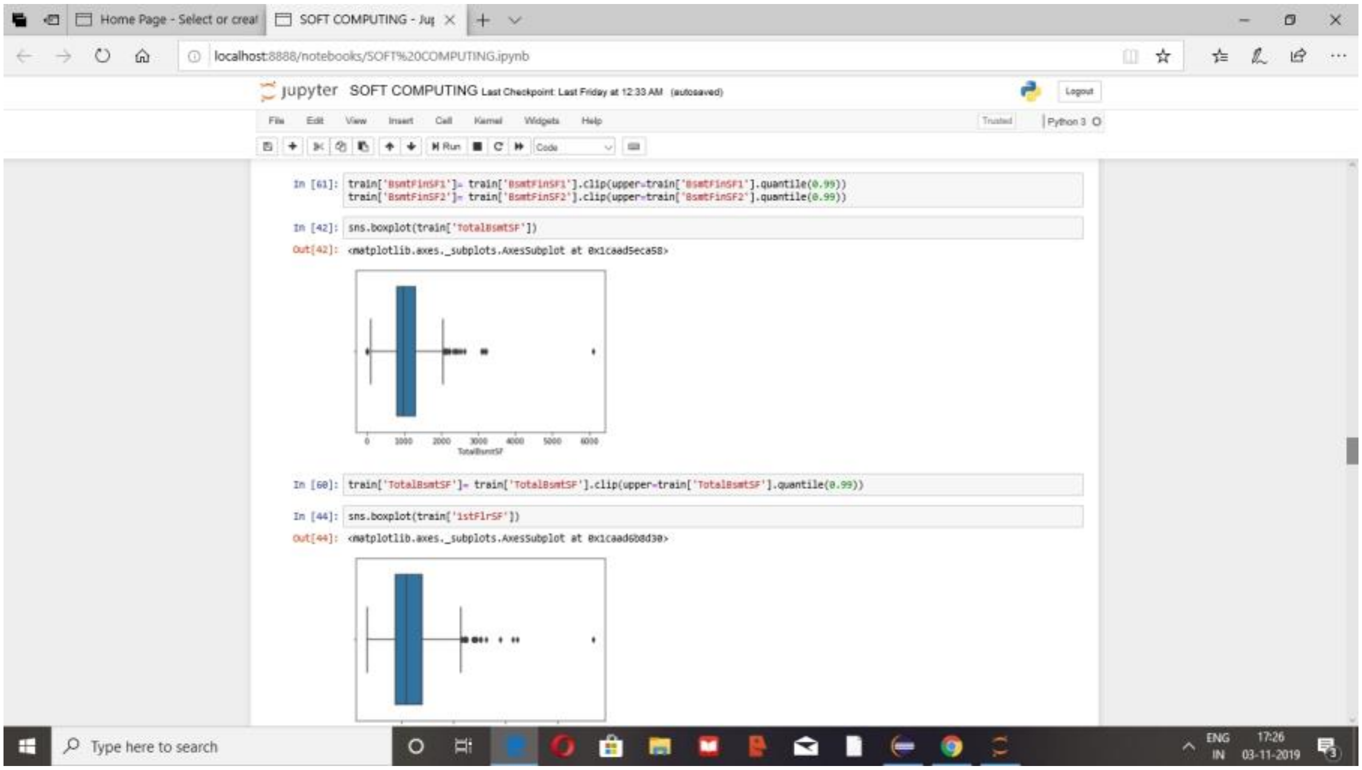
In [33]:
```
sns.boxplot([num_train.LotFrontage])
```

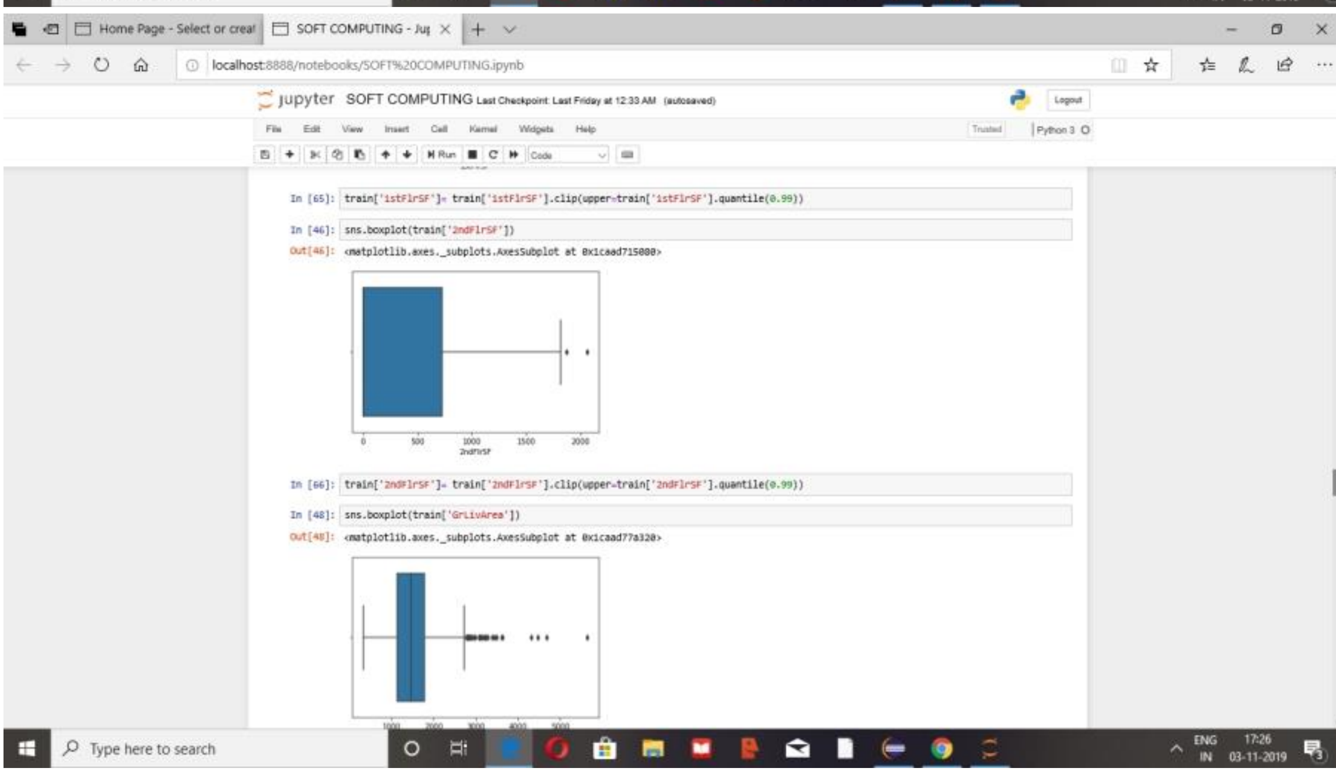
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad0f08d0>

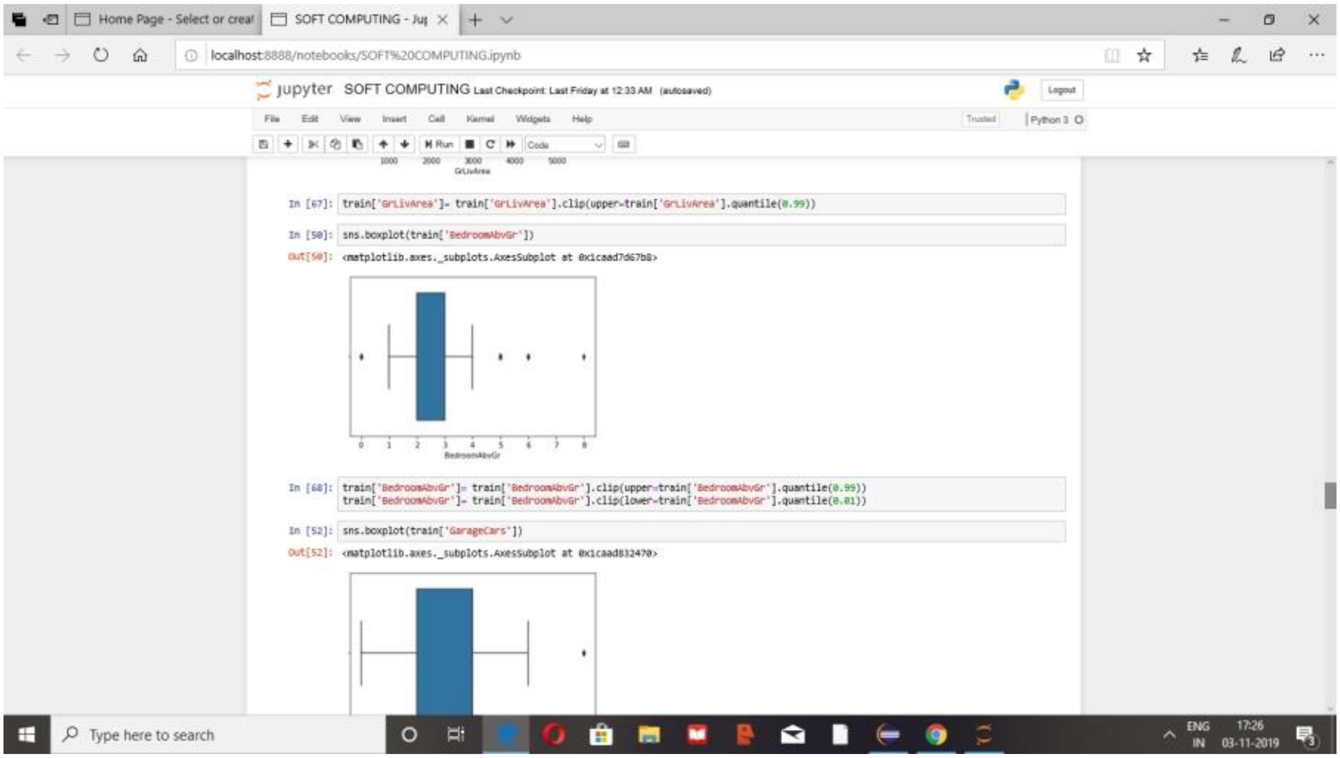



Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [62]: train['LotFrontage']= train['LotFrontage'].clip(upper=train['LotFrontage'].quantile(0.99))
In [35]: sns.boxplot(num_train.LotArea)
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x1caa6351748>
0 50000 100000 150000 200000
LotArea
In [63]: train['LotArea']= train['LotArea'].clip(upper=train['LotArea'].quantile(0.99))
In [37]: sns.boxplot(train['MasVnrArea'])
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad518208>
0 200 400 600 800 1000 1200 1400 1600
Type here to search
ENG IN
17:26
03-11-2019

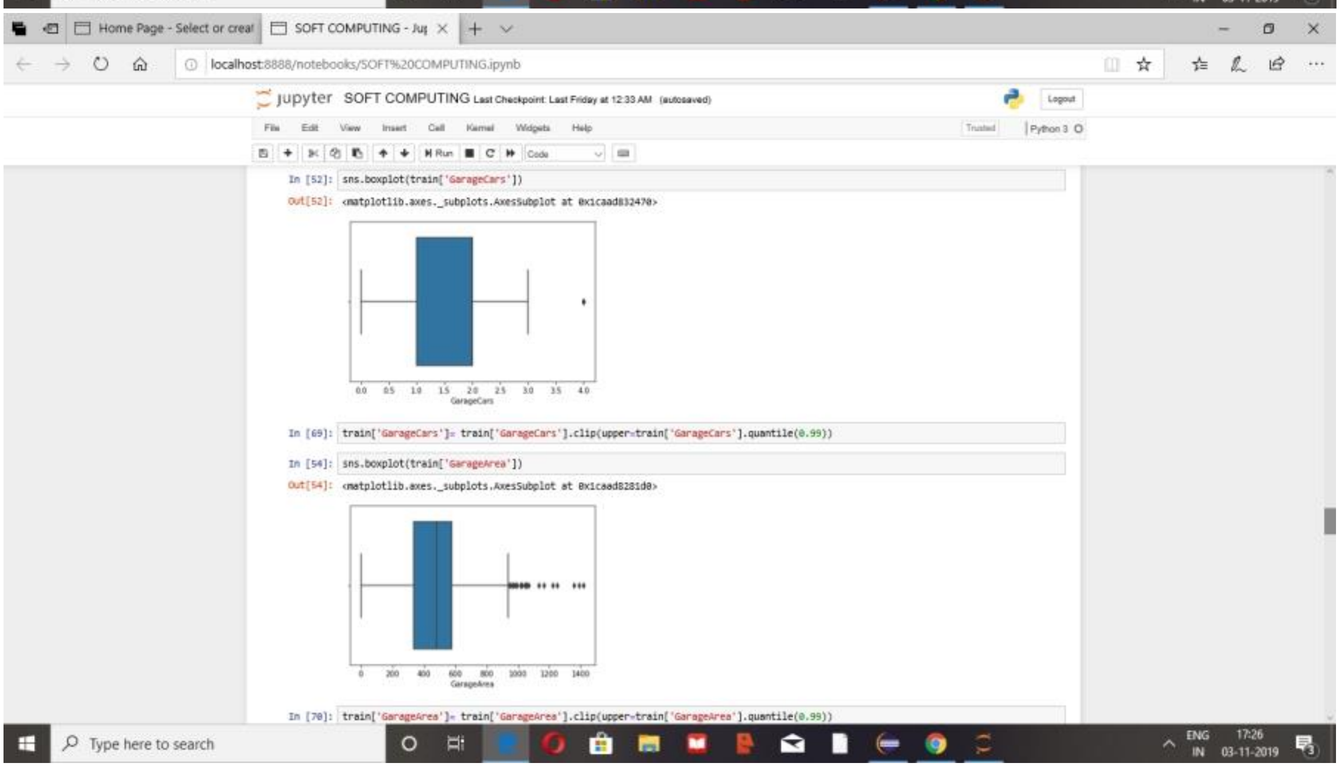

Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [64]: train['MasVnrArea']= train['MasVnrArea'].clip(upper=train['MasVnrArea'].quantile(0.99))
In [39]: sns.boxplot(train['BsmtFinSF1'])
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad551908>
0 1000 2000 3000 4000 5000
BsmtFinSF1
In [40]: sns.boxplot(train['BsmtFinSF2'])
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad5ec4a8>
0 200 400 600 800 1000 1200 1400
BsmtFinSF2
Type here to search
ENG IN
17:26
03-11-2019


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [61]: train['BsmtFinSF1']= train['BsmtFinSF1'].clip(upper=train['BsmtFinSF1'].quantile(0.99))
train['BsmtFinSF2']= train['BsmtFinSF2'].clip(upper=train['BsmtFinSF2'].quantile(0.99))
In [42]: sns.boxplot(train['TotalBsmtSF'])
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad5eca58>
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF
In [60]: train['TotalBsmtSF']= train['TotalBsmtSF'].clip(upper=train['TotalBsmtSF'].quantile(0.99))
In [44]: sns.boxplot(train['1stFlrSF'])
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad6b0d30>
Type here to search
ENG
IN
17:26
03-11-2019

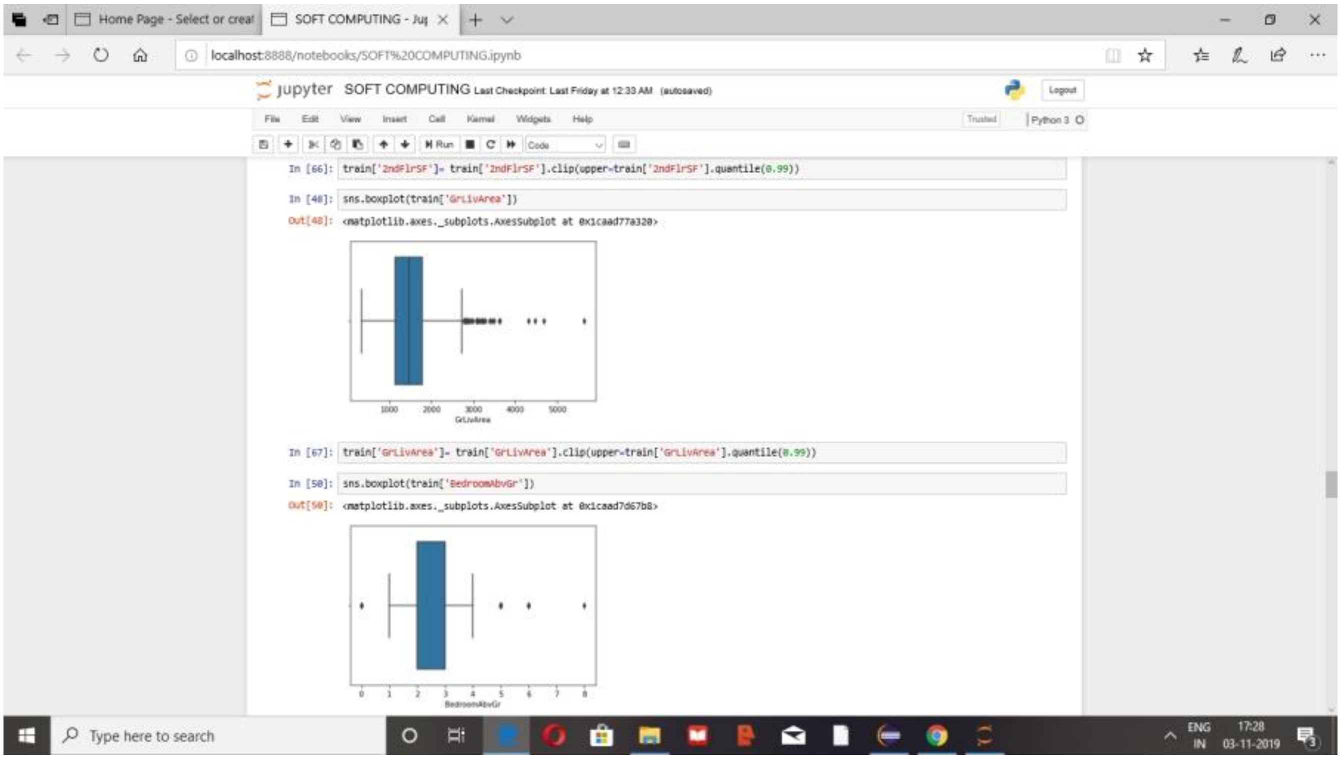

Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [65]: train['1stFlrSF']= train['1stFlrSF'].clip(upper=train['1stFlrSF'].quantile(0.99))
In [46]: sns.boxplot(train['2ndFlrSF'])
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad715080>
0
500
1000
1500
2000
2ndFlrSF
In [66]: train['2ndFlrSF']= train['2ndFlrSF'].clip(upper=train['2ndFlrSF'].quantile(0.99))
In [48]: sns.boxplot(train['GrLivArea'])
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad77a320>
Type here to search
ENG
IN
17:26
03-11-2019



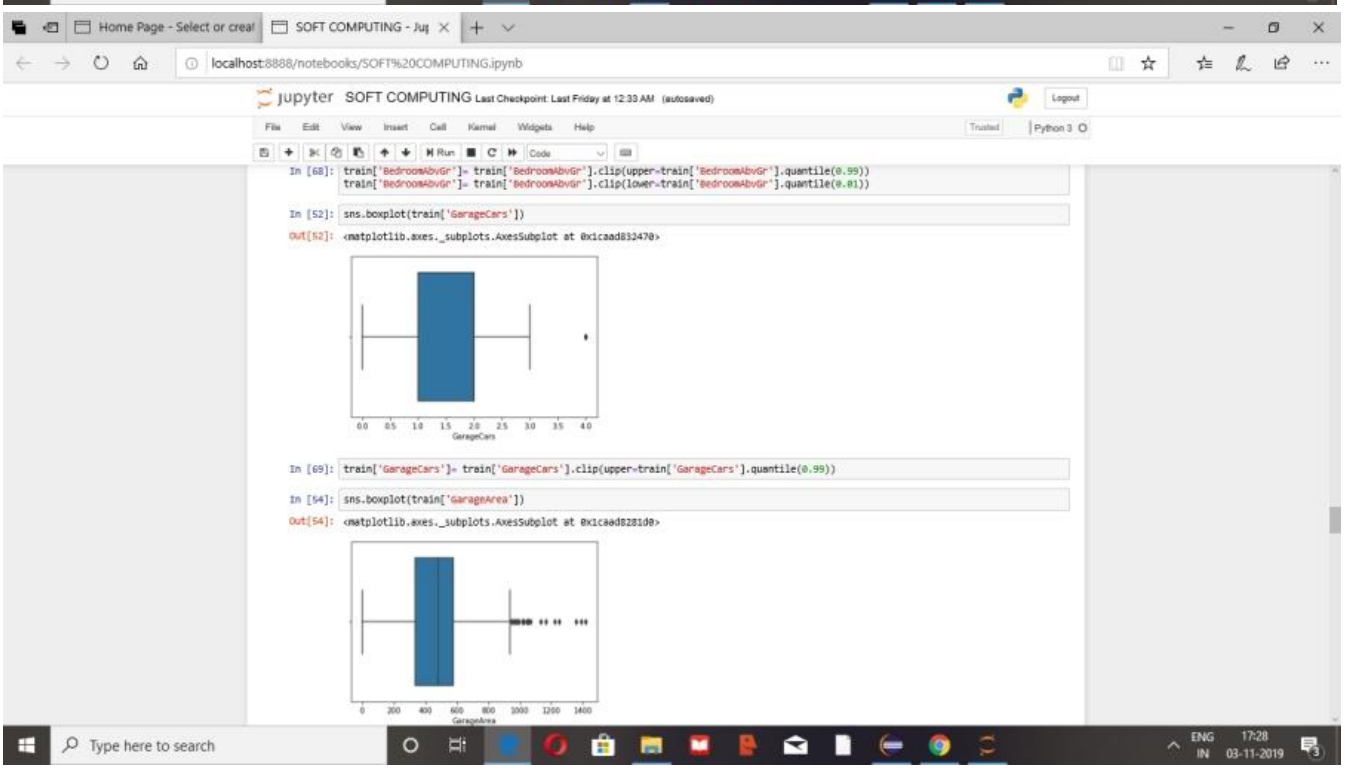
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
In [67]: train['GrLivArea']= train['GrLivArea'].clip(upper=train['GrLivArea'].quantile(0.99))
In [50]: sns.boxplot(train['BedroomAbvGr'])
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad7d67b8>
BedroomAbvGr
In [68]: train['BedroomAbvGr']= train['BedroomAbvGr'].clip(upper=train['BedroomAbvGr'].quantile(0.99))
train['BedroomAbvGr']= train['BedroomAbvGr'].clip(lower=train['BedroomAbvGr'].quantile(0.01))
In [52]: sns.boxplot(train['GarageCars'])
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad832470>
Type here to search



jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
In [52]: sns.boxplot(train['GarageCars'])
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad832470>
GarageCars
In [69]: train['GarageCars']= train['GarageCars'].clip(upper=train['GarageCars'].quantile(0.99))
In [54]: sns.boxplot(train['GarageArea'])
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad8281d0>
GarageArea
In [70]: train['GarageArea']= train['GarageArea'].clip(upper=train['GarageArea'].quantile(0.99))
Type here to search

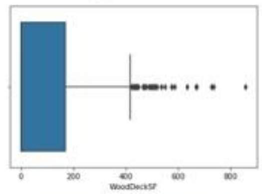


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
In [66]: train['2ndFlrSF']= train['2ndFlrSF'].clip(upper=train['2ndFlrSF'].quantile(0.99))
In [48]: sns.boxplot(train['GrLivArea'])
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad77a320>
1000 2000 3000 4000 5000
GrLivArea
In [67]: train['GrLivArea']= train['GrLivArea'].clip(upper=train['GrLivArea'].quantile(0.99))
In [50]: sns.boxplot(train['BedroomAbvGr'])
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad7d67b8>
0 1 2 3 4 5 6 7 8
BedroomAbvGr
Type here to search
ENG IN 17:28 03-11-2019

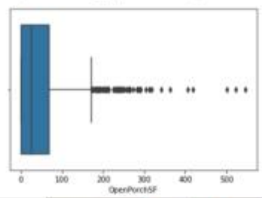


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
In [68]: train['BedroomAbvGr']= train['BedroomAbvGr'].clip(upper=train['BedroomAbvGr'].quantile(0.99))
train['BedroomAbvGr']= train['BedroomAbvGr'].clip(lower=train['BedroomAbvGr'].quantile(0.01))
In [52]: sns.boxplot(train['GarageCars'])
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad832470>
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
GarageCars
In [69]: train['GarageCars']= train['GarageCars'].clip(upper=train['GarageCars'].quantile(0.99))
In [54]: sns.boxplot(train['GarageArea'])
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad8281d0>
0 200 400 600 800 1000 1200 1400
GarageArea
Type here to search
ENG IN 17:28 03-11-2019

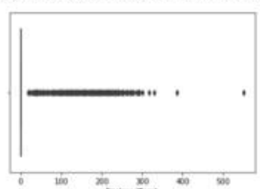
```
In [70]: train['GarageArea']= train['GarageArea'].clip(upper=train['GarageArea'].quantile(0.99))

In [56]: sns.boxplot(train['WoodDeckSF'])
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad907e48>
```



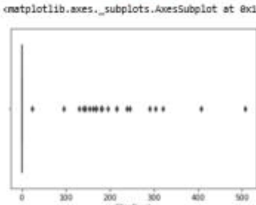

```
In [71]: train['WoodDeckSF']= train['WoodDeckSF'].clip(upper=train['WoodDeckSF'].quantile(0.99))

In [58]: sns.boxplot(train['OpenPorchSF'])
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad96f7b8>
```

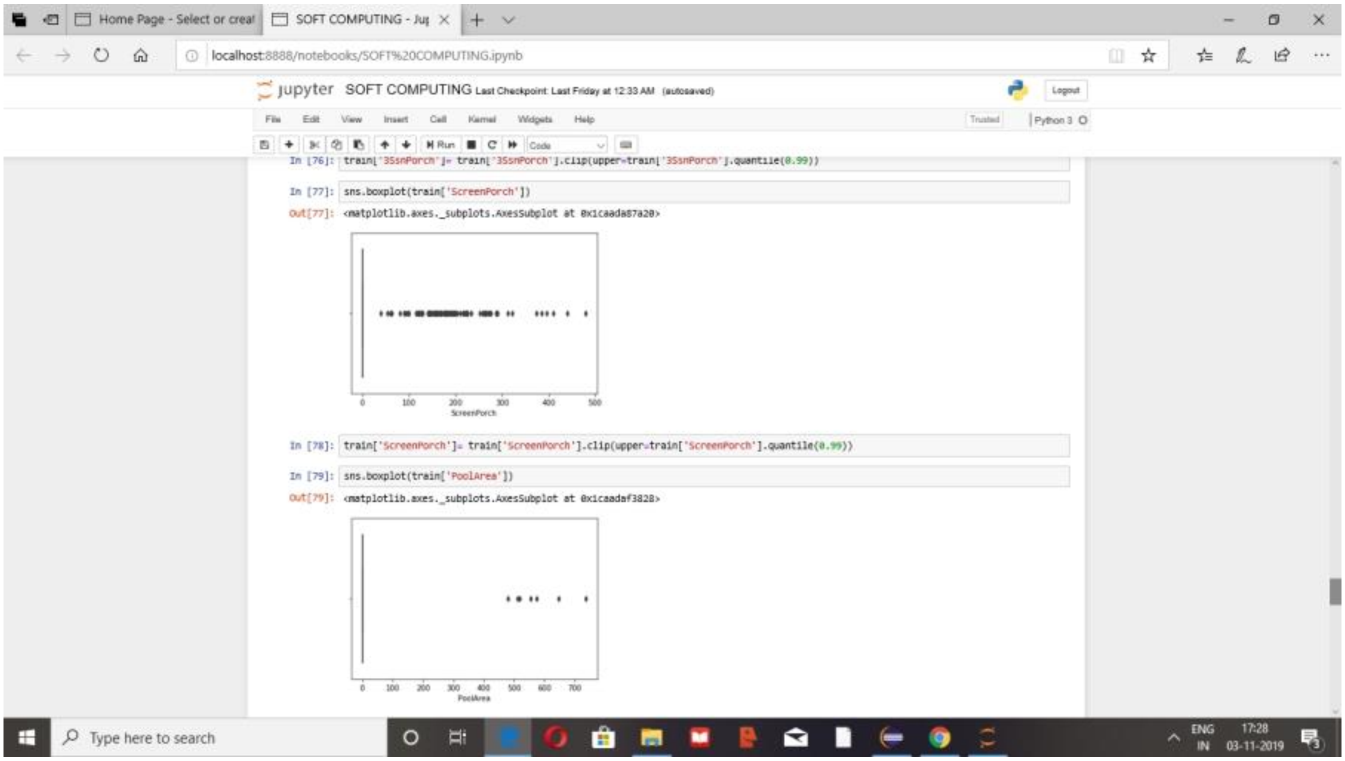


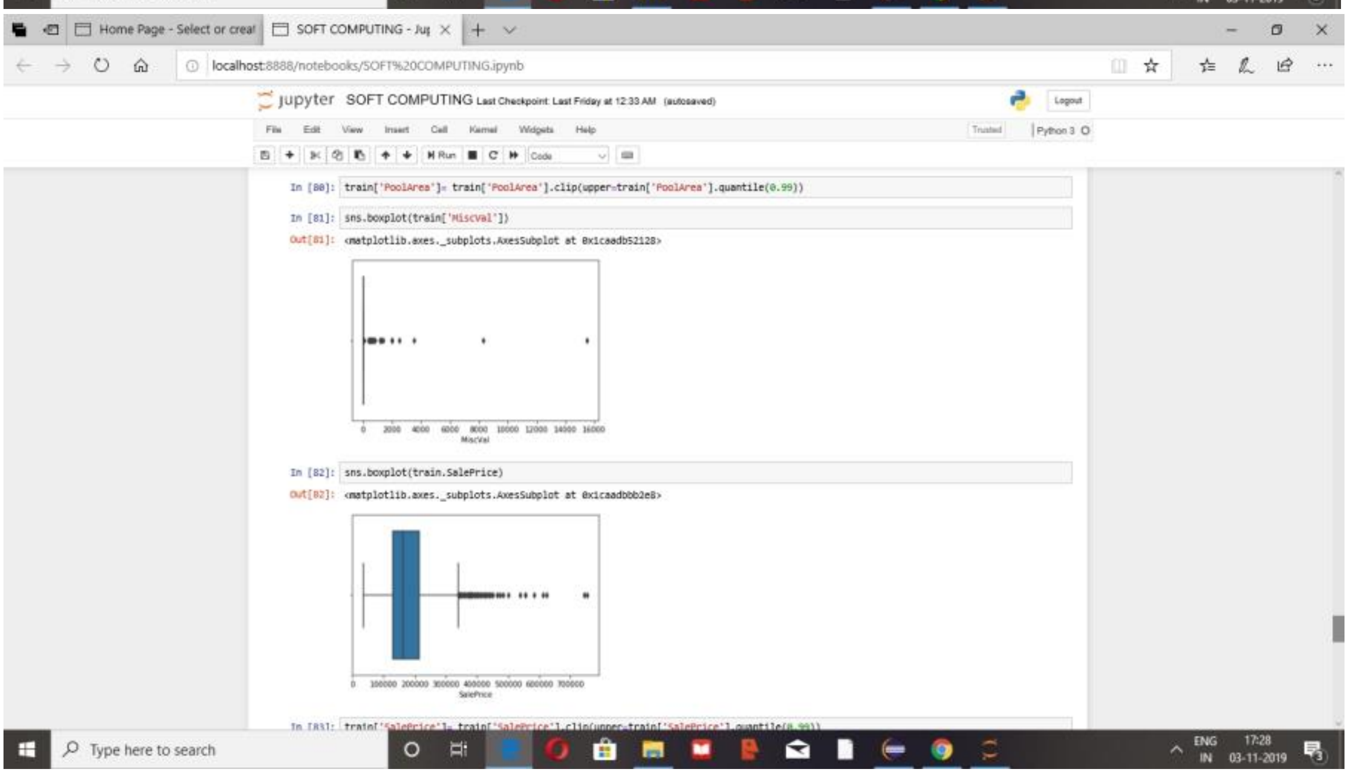

```
In [72]: train['OpenPorchSF']= train['OpenPorchSF'].clip(upper=train['OpenPorchSF'].quantile(0.99))

In [73]: sns.boxplot(train['EnclosedPorch'])
Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1caad9b8f98>
```

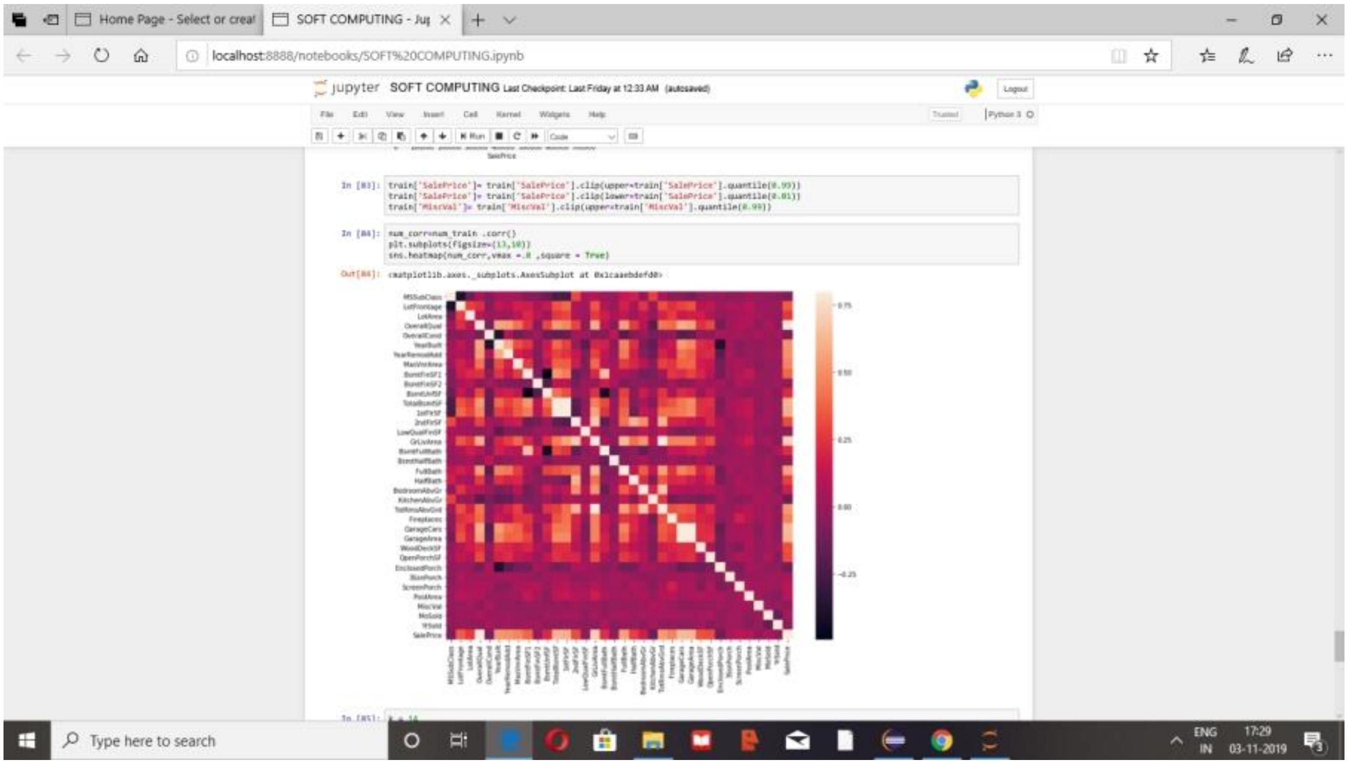


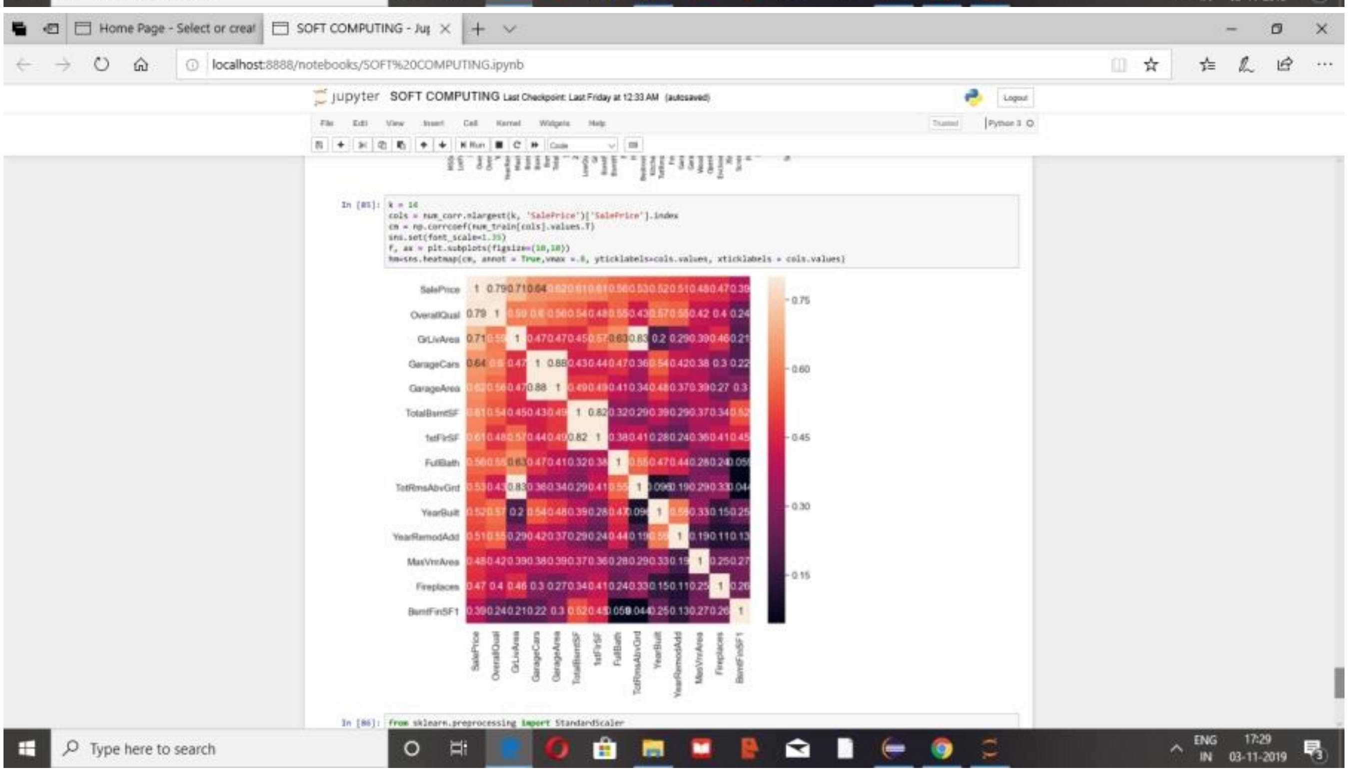

```
In [74]: train['EnclosedPorch']= train['EnclosedPorch'].clip(upper=train['EnclosedPorch'].quantile(0.99))

In [75]: sns.boxplot(train['3SsnPorch'])
Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x1caada1fda0>
```

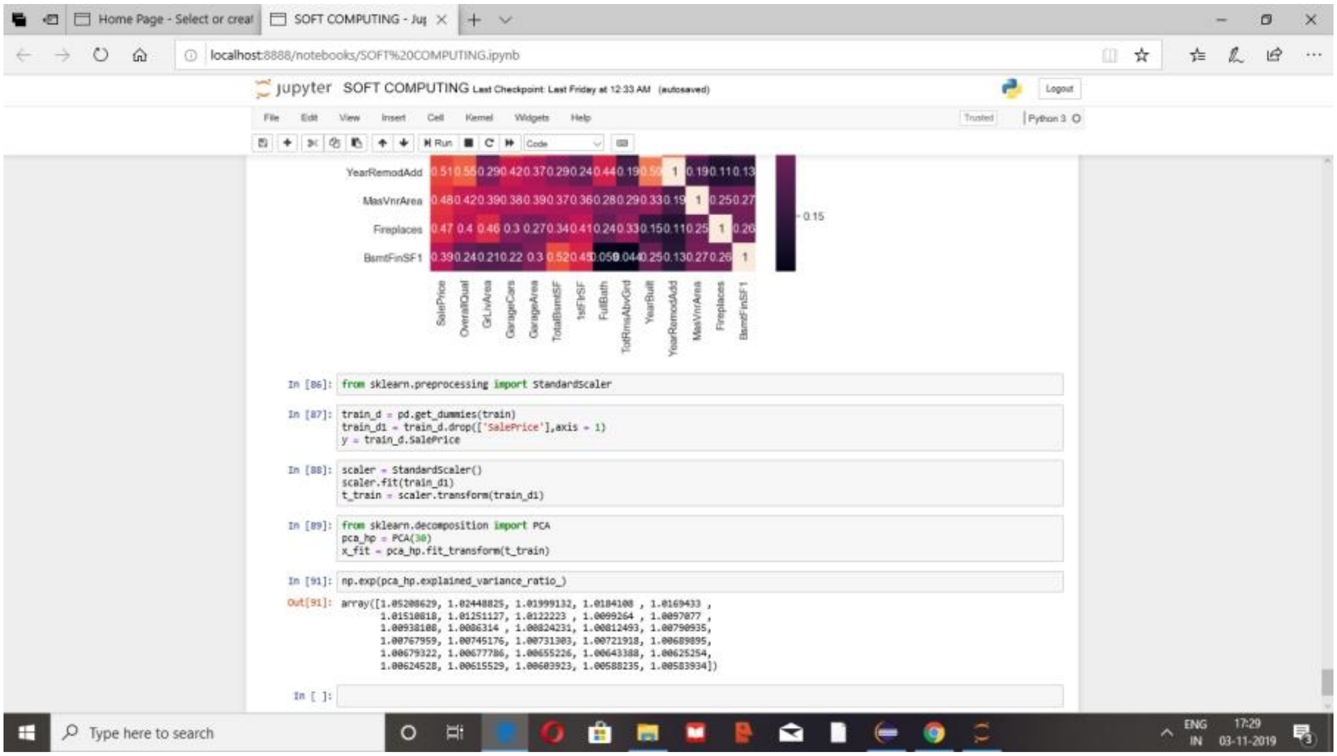



Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [76]: train['3SsnPorch']= train['3SsnPorch'].clip(upper=train['3SsnPorch'].quantile(0.99))
In [77]: sns.boxplot(train['ScreenPorch'])
Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1caada87a20>
0 100 200 300 400 500
ScreenPorch
In [78]: train['ScreenPorch']= train['ScreenPorch'].clip(upper=train['ScreenPorch'].quantile(0.99))
In [79]: sns.boxplot(train['PoolArea'])
Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0x1caadaf3828>
0 100 200 300 400 500 600 700
PoolArea
Type here to search
ENG IN 17:28 03-11-2019


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [80]: train['PoolArea']= train['PoolArea'].clip(upper=train['PoolArea'].quantile(0.99))
In [81]: sns.boxplot(train['MiscVal'])
Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x1caadb52128>
0 2000 4000 6000 8000 10000 12000 14000 16000
MiscVal
In [82]: sns.boxplot(train.SalePrice)
Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x1caadbbb2e8>
0 100000 200000 300000 400000 500000 600000 700000
SalePrice
Type here to search
ENG IN 17:28 03-11-2019


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [83]:
train['SalePrice']= train['SalePrice'].clip(upper=train['SalePrice'].quantile(0.99))
train['SalePrice']= train['SalePrice'].clip(lower=train['SalePrice'].quantile(0.01))
train['MiscVal']= train['MiscVal'].clip(upper=train['MiscVal'].quantile(0.99))
In [84]:
num_corr=num_train .corr()
plt.subplots(figsize=(13,10))
sns.heatmap(num_corr,vmax =.8 ,square = True)
Out[84]: <matplotlib.axes._subplots.AxesSubplot at 0x1caaebdefd0>
Type here to search
ENG
IN
17:29
03-11-2019


Home Page - Select or creat
SOFT COMPUTING - Jup
localhost:8888/notebooks/SOFT%20COMPUTING.ipynb
jupyter SOFT COMPUTING Last Checkpoint: Last Friday at 12:33 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [85]:
k = 14
cols = num_corr.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(num_train[cols].values.T)
sns.set(font_scale=1.35)
f, ax = plt.subplots(figsize=(10,10))
hm=sns.heatmap(cm, annot = True,vmax =.8, yticklabels=cols.values, xticklabels = cols.values)
SalePrice
OverallQual
GrLivArea
GarageCars
GarageArea
TotalBsmtSF
1stFlrSF
FullBath
TotRmsAbvGrd
YearBuilt
YearRemodAdd
MasVnrArea
Fireplaces
BsmtFinSF1
0.75
0.60
0.45
0.30
0.15
In [86]: from sklearn.preprocessing import StandardScaler
Type here to search
ENG
IN
17:29
03-11-2019



- ***<u>nn code:</u>***

```
import os

import keras

from __future__ import absolute_import

from __future__ import division

from __future__ import print_function


import itertools


import pandas as pd
```

```
import numpy as np

import matplotlib.pyplot as plt

from pylab import rcParams

import matplotlib


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler

train = pd.read_csv(r'C:\Users\Nithishma\Desktop\train.csv')

print('Shape of the train data with all features:', train.shape)

train = train.select_dtypes(exclude=['object'])

print("")

print('Shape of the train data with numerical features:', train.shape)

train.drop('Id',axis = 1, inplace = True)

train.fillna(0,inplace=True)

test = pd.read_csv(r'C:\Users\Nithishma\Desktop\test.csv')

test = test.select_dtypes(exclude=['object'])

ID = test.Id

test.fillna(0,inplace=True)

test.drop('Id',axis = 1, inplace = True)


print("")

print("List of features contained our dataset:",list(train.columns))
```

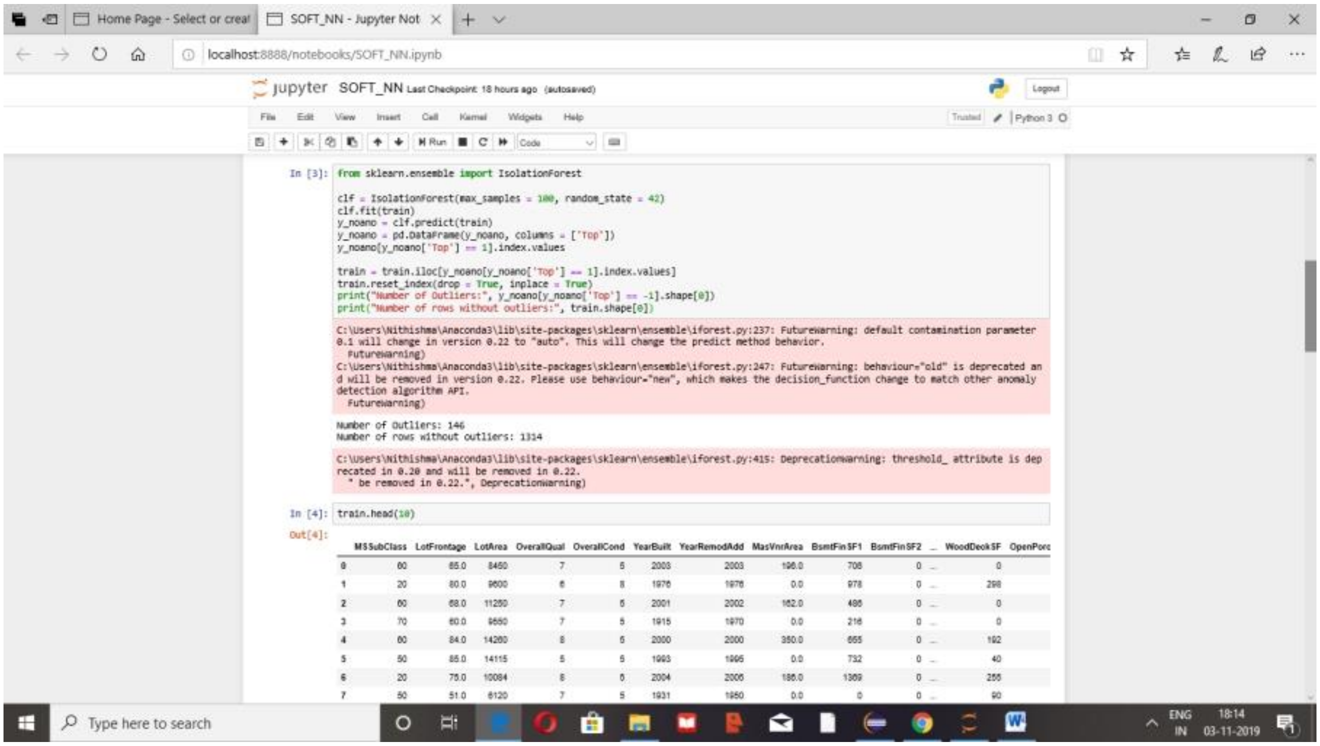
```
from sklearn.ensemble import IsolationForest


clf = IsolationForest(max_samples = 100, random_state = 42)
clf.fit(train)
y_noano = clf.predict(train)
y_noano = pd.DataFrame(y_noano, columns = ['Top'])
y_noano[y_noano['Top'] == 1].index.values


train = train.iloc[y_noano[y_noano['Top'] == 1].index.values]
train.reset_index(drop = True, inplace = True)
print("Number of Outliers:", y_noano[y_noano['Top'] == -1].shape[0])
print("Number of rows without outliers:", train.shape[0])
import warnings
warnings.filterwarnings('ignore')


col_train = list(train.columns)
col_train_bis = list(train.columns)


col_train_bis.remove('SalePrice')


mat_train = np.matrix(train)
mat_test  = np.matrix(test)
```

```
mat_new = np.matrix(train.drop('SalePrice',axis = 1))

mat_y = np.array(train.SalePrice).reshape((1314,1))


prepro_y = MinMaxScaler()

prepro_y.fit(mat_y)


prepro = MinMaxScaler()

prepro.fit(mat_train)


prepro_test = MinMaxScaler()

prepro_test.fit(mat_new)


train = pd.DataFrame(prepro.transform(mat_train),columns = col_train)

test  = pd.DataFrame(prepro_test.transform(mat_test),columns =
col_train_bis)

train.head()

COLUMNS = col_train

FEATURES = col_train_bis

LABEL = "SalePrice"


# Columns

feature_cols = FEATURES
```

```
# Training set and Prediction set with the features to predict

training_set = train[COLUMNS]

prediction_set = train.SalePrice


# Train and Test

x_train, x_test, y_train, y_test = train_test_split(training_set[FEATURES] ,
prediction_set, test_size=0.33, random_state=42)

y_train = pd.DataFrame(y_train, columns = [LABEL])

training_set = pd.DataFrame(x_train, columns = FEATURES).merge(y_train,
left_index = True, right_index = True)

training_set.head()


# Training for submission

training_sub = training_set[col_train]

y_test = pd.DataFrame(y_test, columns = [LABEL])

testing_set = pd.DataFrame(x_test, columns = FEATURES).merge(y_test,
left_index = True, right_index = True)

testing_set.head()

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit_learn import KerasRegressor
```

```
seed = 7

np.random.seed(seed)


# Model

model = Sequential()

model.add(Dense(200, input_dim=36, kernel_initializer='normal',
activation='relu'))

model.add(Dense(100, kernel_initializer='normal', activation='relu'))

model.add(Dense(50, kernel_initializer='normal', activation='relu'))

model.add(Dense(25, kernel_initializer='normal', activation='relu'))

model.add(Dense(1, kernel_initializer='normal'))

# Compile model

model.compile(loss='mean_squared_error',
optimizer=keras.optimizers.Adadelta())


feature_cols = training_set[FEATURES]

labels = training_set[LABEL].values


model.fit(np.array(feature_cols), np.array(labels), epochs=100, batch_size=10)

# Evaluation on the test set created by train_test_split

model.evaluate(np.array(feature_cols), np.array(labels))
```

```
feature_cols_test = testing_set[FEATURES]

labels_test = testing_set[LABEL].values


y = model.predict(np.array(feature_cols_test))

predictions = list(itertools.islice(y, testing_set.shape[0]))

predictions =
prepro_y.inverse_transform(np.array(predictions).reshape(434,1))

reality = pd.DataFrame(prepro.inverse_transform(testing_set), columns =
[COLUMNS]).SalePrice

y_predict = model.predict(np.array(test))


def to_submit(pred_y,name_out):

    y_predict = list(itertools.islice(pred_y, test.shape[0]))

y_predict=pd.DataFrame(prepro_y.inverse_transform(np.array(y_predict).resh
ape(len(y_predict),1)), columns = ['SalePrice'])

    y_predict = y_predict.join(ID)

    y_predict.to_csv(name_out + '.csv',index=False)


to_submit(y_predict, "Realestate_prices")
```



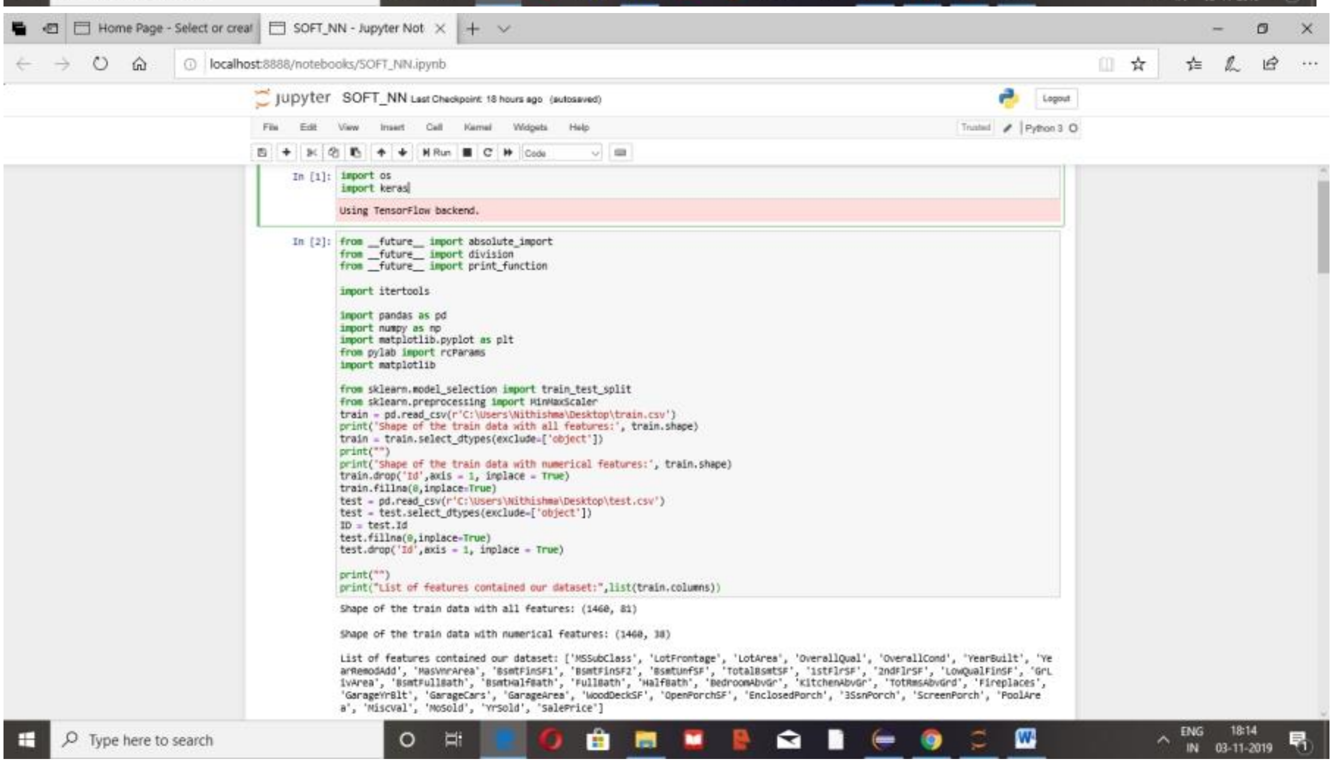
In [3]:
```
from sklearn.ensemble import IsolationForest

clf = IsolationForest(max_samples = 100, random_state = 42)
clf.fit(train)
y_noano = clf.predict(train)
y_noano = pd.DataFrame(y_noano, columns = ['Top'])
y_noano[y_noano['Top'] == 1].index.values

train = train.iloc[y_noano[y_noano['Top'] == 1].index.values]
train.reset_index(drop = True, inplace = True)
print("Number of Outliers:", y_noano[y_noano['Top'] == -1].shape[0])
print("Number of rows without outliers:", train.shape[0])
```

```
C:\Users\Nithishma\Anaconda3\lib\site-packages\sklearn\ensemble\iforest.py:237: FutureWarning: default contamination parameter 0.1 will change in version 0.22 to "auto". This will change the predict method behavior.
  FutureWarning)
C:\Users\Nithishma\Anaconda3\lib\site-packages\sklearn\ensemble\iforest.py:247: FutureWarning: behaviour="old" is deprecated and will be removed in version 0.22. Please use behaviour="new", which makes the decision_function change to match other anomaly detection algorithm API.
  FutureWarning)

Number of Outliers: 146
Number of rows without outliers: 1314

C:\Users\Nithishma\Anaconda3\lib\site-packages\sklearn\ensemble\iforest.py:415: DeprecationWarning: threshold_ attribute is deprecated in 0.20 and will be removed in 0.22.
  " be removed in 0.22.", DeprecationWarning)
```

In [4]:
```
train.head(10)
```

Out[4]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeckSF	OpenPorc
0	60	65.0	8450	7	5	2003	2003	196.0	706	0	...	0	
1	20	80.0	9600	6	8	1976	1976	0.0	978	0	...	298	
2	60	68.0	11250	7	5	2001	2002	162.0	486	0	...	0	
3	70	60.0	9550	7	5	1915	1970	0.0	216	0	...	0	
4	60	84.0	14260	8	5	2000	2000	350.0	655	0	...	192	
5	50	85.0	14115	5	5	1993	1995	0.0	732	0	...	40	
6	20	75.0	10084	8	5	2004	2005	186.0	1369	0	...	255	
7	50	51.0	6120	7	5	1931	1950	0.0	0	0	...	90	



In [1]:
```
import os
import keras

Using TensorFlow backend.
```

In [2]:
```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import itertools

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
import matplotlib

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
train = pd.read_csv(r'C:\Users\Nithishma\Desktop\train.csv')
print('Shape of the train data with all features:', train.shape)
train = train.select_dtypes(exclude=['object'])
print("")
print('Shape of the train data with numerical features:', train.shape)
train.drop('Id',axis = 1, inplace = True)
train.fillna(0,inplace=True)
test = pd.read_csv(r'C:\Users\Nithishma\Desktop\test.csv')
test = test.select_dtypes(exclude=['object'])
ID = test.Id
test.fillna(0,inplace=True)
test.drop('Id',axis = 1, inplace = True)

print("")
print("List of features contained our dataset:",list(train.columns))
```

```
Shape of the train data with all features: (1460, 81)

Shape of the train data with numerical features: (1460, 38)

List of features contained our dataset: ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice']
```

localhost:8888/notebooks/SOFT_NN.ipynb

jupyter SOFT_NN Last Checkpoint: 18 hours ago (autosaved)

In [5]:
```python
import warnings
warnings.filterwarnings('ignore')

col_train = list(train.columns)
col_train_bis = list(train.columns)

col_train_bis.remove('SalePrice')

mat_train = np.matrix(train)
mat_test  = np.matrix(test)
mat_new = np.matrix(train.drop('SalePrice',axis = 1))
mat_y = np.array(train.SalePrice).reshape((1314,1))

prepro_y = MinMaxScaler()
prepro_y.fit(mat_y)

prepro = MinMaxScaler()
prepro.fit(mat_train)

prepro_test = MinMaxScaler()
prepro_test.fit(mat_new)

train = pd.DataFrame(prepro.transform(mat_train),columns = col_train)
test  = pd.DataFrame(prepro_test.transform(mat_test),columns = col_train_bis)
train.head()
```

Out[5]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeckSF	OpenPor
0	0.235294	0.207668	0.082802	0.625	0.428571	0.953488	0.883333	0.12250	0.416274	0.0	...	0.000000	0.16
1	0.000000	0.255591	0.072904	0.500	0.857143	0.744186	0.433333	0.00000	0.578651	0.0	...	0.404891	0.00
2	0.235294	0.217252	0.087396	0.625	0.428571	0.937984	0.866667	0.10125	0.286557	0.0	...	0.000000	0.11
3	0.294118	0.191693	0.072464	0.625	0.428571	0.271318	0.333333	0.00000	0.127358	0.0	...	0.000000	0.00
4	0.235294	0.268371	0.113835	0.750	0.428571	0.930233	0.833333	0.21875	0.386203	0.0	...	0.260870	0.22

5 rows × 37 columns

In [6]:
```python
COLUMNS = col_train
FEATURES = col_train_bis
LABEL = "SalePrice"

# Columns
feature_cols = FEATURES

# Training set and Prediction set with the features to predict
training_set = train[COLUMNS]
prediction_set = train.SalePrice

# Train and Test
x_train, x_test, y_train, y_test = train_test_split(training_set[FEATURES] , prediction_set, test_size=0.33, random_state=42)
y_train = pd.DataFrame(y_train, columns = [LABEL])
training_set = pd.DataFrame(x_train, columns = FEATURES).merge(y_train, left_index = True, right_index = True)
training_set.head()

# Training for submission
training_sub = training_set[col_train]
```

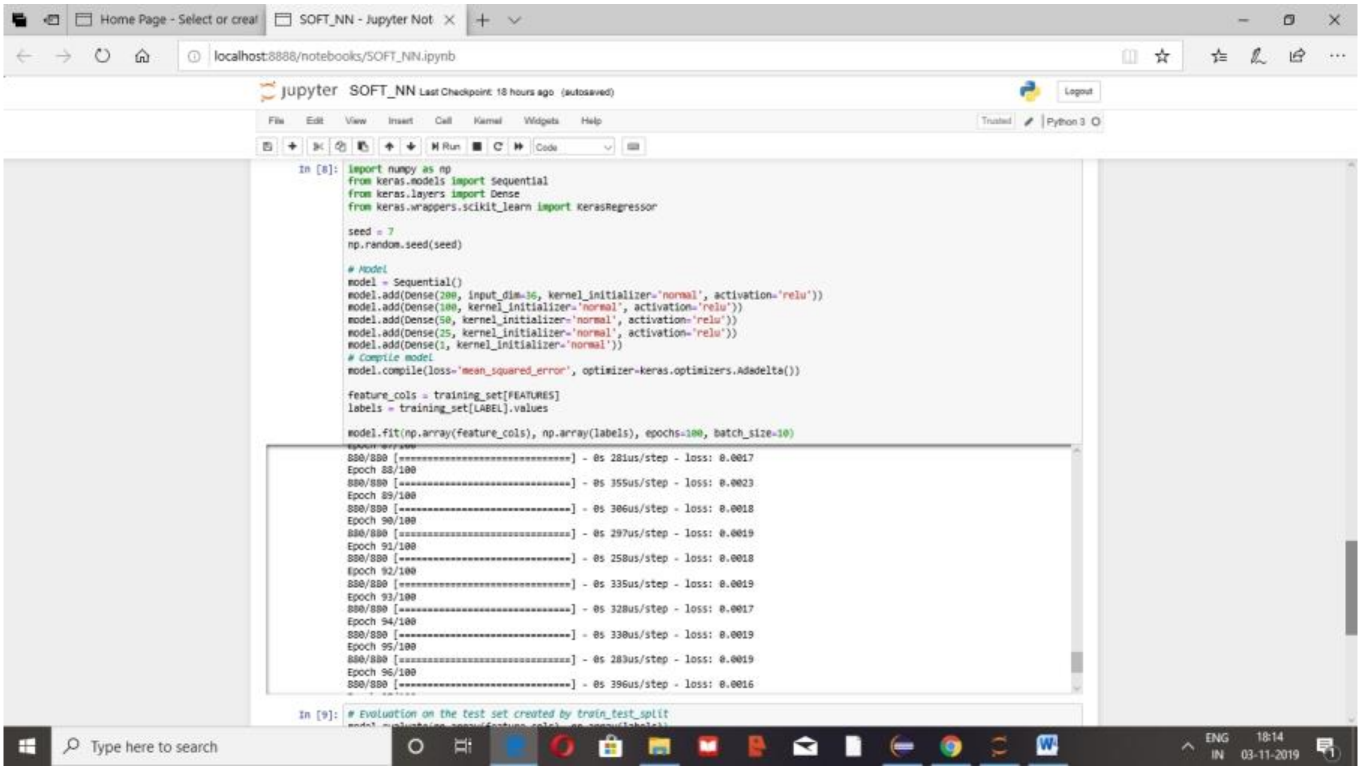
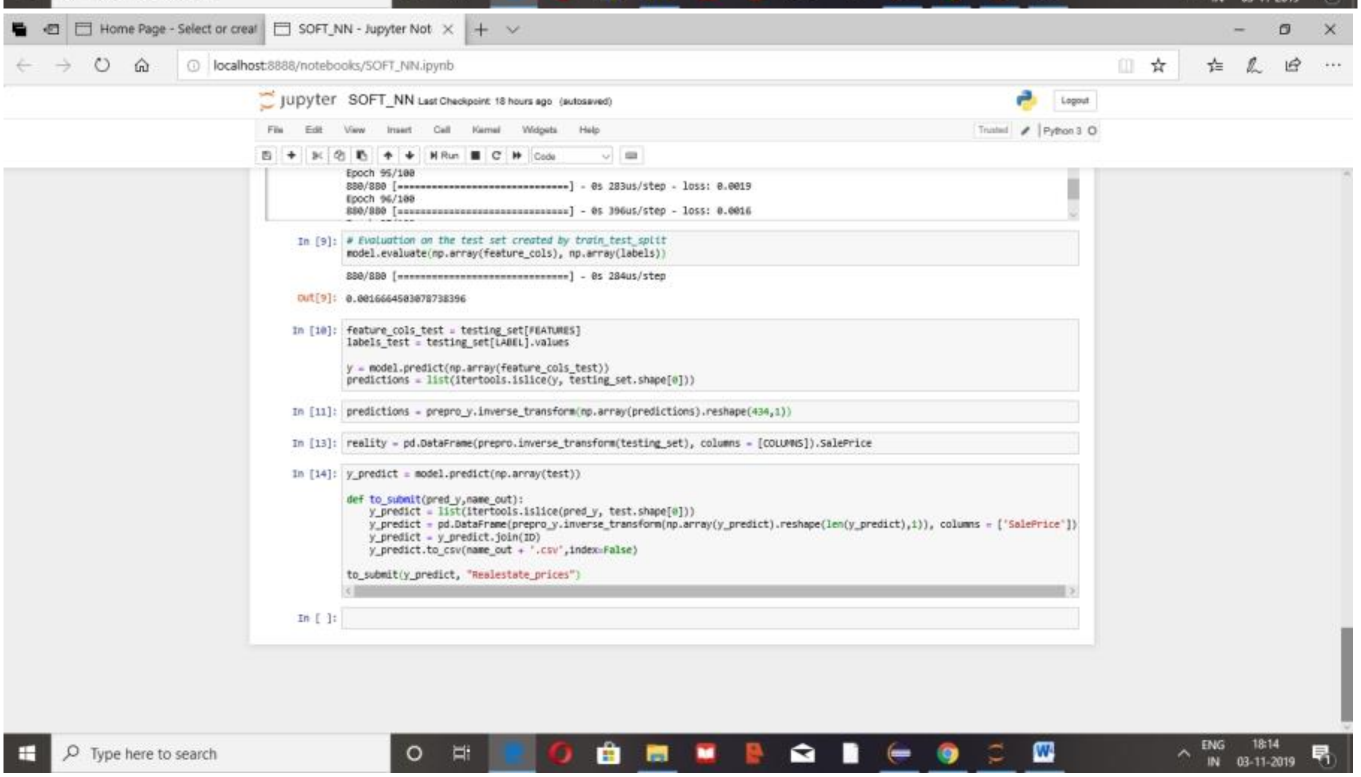
In [7]:
```python
y_test = pd.DataFrame(y_test, columns = [LABEL])
testing_set = pd.DataFrame(x_test, columns = FEATURES).merge(y_test, left_index = True, right_index = True)
testing_set.head()
```

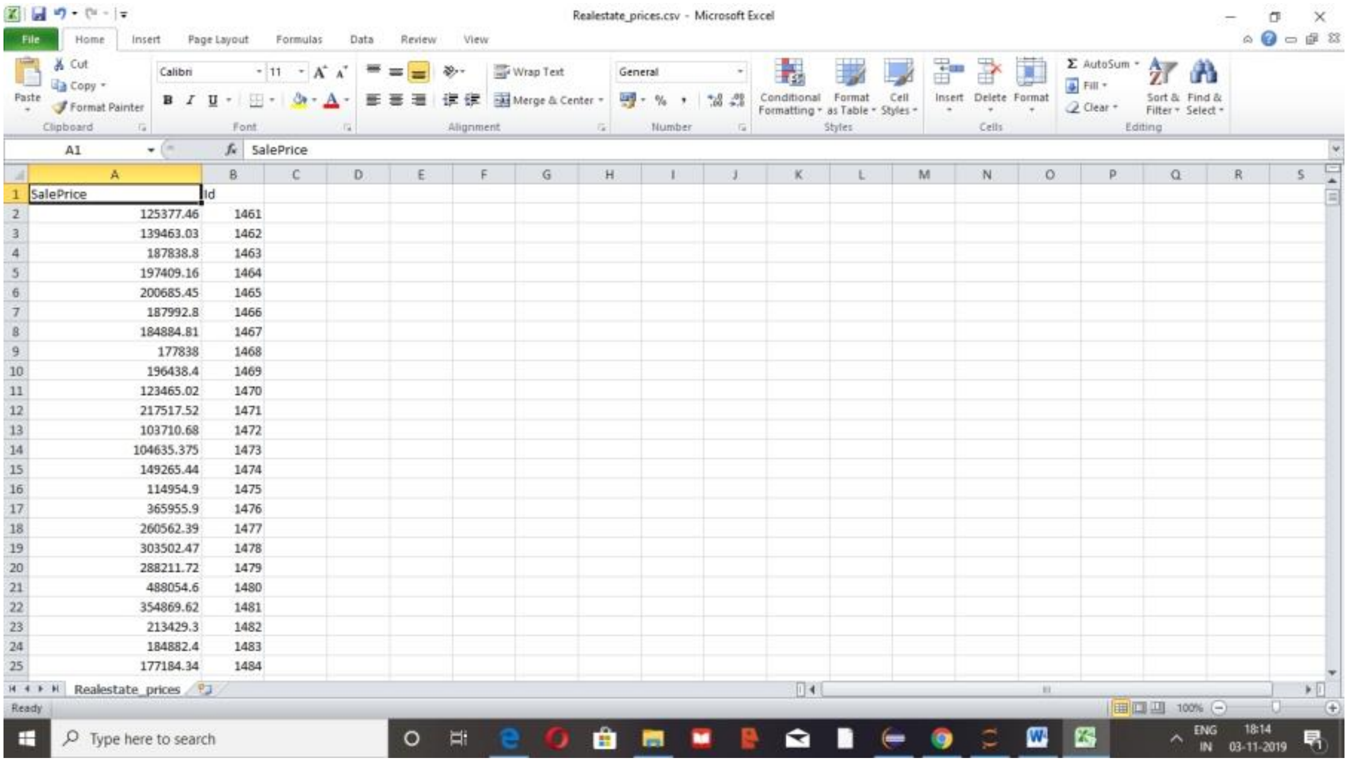
Out[7]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeckSF	Open
1232	0.235294	0.236617	0.074221	0.625	0.571429	0.914729	0.800000	0.00000	0.574882	0.0	...	0.343750	
76	0.352941	0.000000	0.083505	0.625	0.428571	0.891473	0.766667	0.01375	0.000000	0.0	...	0.163043	
962	0.235294	0.249201	0.091349	0.500	0.571429	0.682171	0.300000	0.00000	0.175708	0.0	...	0.000000	
433	0.294118	0.159744	0.010540	0.625	0.857143	0.271318	0.916667	0.00000	0.176297	0.0	...	0.000000	
1110	0.294118	0.306709	0.103927	0.375	0.428571	0.263566	0.000000	0.00000	0.000000	0.0	...	0.275815	

5 rows × 37 columns

In [8]:
```python
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
```


Home Page - Select or creat
SOFT_NN - Jupyter Not
localhost:8888/notebooks/SOFT_NN.ipynb
jupyter SOFT_NN Last Checkpoint: 18 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Logout
In [8]:
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
seed = 7
np.random.seed(seed)
# Model
model = Sequential()
model.add(Dense(200, input_dim=36, kernel_initializer='normal', activation='relu'))
model.add(Dense(100, kernel_initializer='normal', activation='relu'))
model.add(Dense(50, kernel_initializer='normal', activation='relu'))
model.add(Dense(25, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
# Compile model
model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adadelta())
feature_cols = training_set[FEATURES]
labels = training_set[LABEL].values
model.fit(np.array(feature_cols), np.array(labels), epochs=100, batch_size=10)
880/880 [==============================] - 0s 281us/step - loss: 0.0017
Epoch 88/100
880/880 [==============================] - 0s 355us/step - loss: 0.0023
Epoch 89/100
880/880 [==============================] - 0s 306us/step - loss: 0.0018
Epoch 90/100
880/880 [==============================] - 0s 297us/step - loss: 0.0019
Epoch 91/100
880/880 [==============================] - 0s 258us/step - loss: 0.0018
Epoch 92/100
880/880 [==============================] - 0s 335us/step - loss: 0.0019
Epoch 93/100
880/880 [==============================] - 0s 328us/step - loss: 0.0017
Epoch 94/100
880/880 [==============================] - 0s 330us/step - loss: 0.0019
Epoch 95/100
880/880 [==============================] - 0s 283us/step - loss: 0.0019
Epoch 96/100
880/880 [==============================] - 0s 396us/step - loss: 0.0016
In [9]: # Evaluation on the test set created by train_test_split
Type here to search
ENG IN 18:14 03-11-2019

Home Page - Select or creat
SOFT_NN - Jupyter Not
localhost:8888/notebooks/SOFT_NN.ipynb
jupyter SOFT_NN Last Checkpoint: 18 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Logout
Epoch 95/100
880/880 [==============================] - 0s 283us/step - loss: 0.0019
Epoch 96/100
880/880 [==============================] - 0s 396us/step - loss: 0.0016
In [9]:
# Evaluation on the test set created by train_test_split
model.evaluate(np.array(feature_cols), np.array(labels))
880/880 [==============================] - 0s 284us/step
Out[9]: 0.0016664503078738396
In [10]:
feature_cols_test = testing_set[FEATURES]
labels_test = testing_set[LABEL].values
y = model.predict(np.array(feature_cols_test))
predictions = list(itertools.islice(y, testing_set.shape[0]))
In [11]: predictions = prepro_y.inverse_transform(np.array(predictions).reshape(434,1))
In [13]: reality = pd.DataFrame(prepro.inverse_transform(testing_set), columns = [COLUMNS]).SalePrice
In [14]:
y_predict = model.predict(np.array(test))
def to_submit(pred_y,name_out):
y_predict = list(itertools.islice(pred_y, test.shape[0]))
y_predict = pd.DataFrame(prepro_y.inverse_transform(np.array(y_predict).reshape(len(y_predict),1)), columns = ['SalePrice'])
y_predict = y_predict.join(ID)
y_predict.to_csv(name_out + '.csv',index=False)
to_submit(y_predict, "Realestate_prices")
In [ ]:
Type here to search
ENG IN 18:14 03-11-2019



COMPARATIVE STUDY :

The use of the neural network model is similar to the process utilized in building the hedonic price model. However, the neural network must first be trained from a set of data. For a particular input, an output (estimated house price) is produced from the model. Then, the model compares the model output to the actual output (actual house price). The accuracy of this value is determined by the total mean square error and then back propagation is used in an attempt to reduce prediction errors, which is done through the adjusting of the connection weights.

The performance of the network can be influenced by the number of hidden layers and the number of nodes that are included in each hidden layer. Unfortunately, there exists little theory to support the process for the determination of the optimal number of hidden layers and nodes, and also the optimal internal error threshold (Lenk et al., 1997). Therefore, a trial-and-error process is applied to find the optimal artificial neural network model. A feed-forward/back-propagation neural network software package, NeuroShell, was used to construct the artificial neural network model.

CONCLUSION AND FUTURE WORK:

This project presented the development of an artificial neural network-based model that is designed to support real estate investors and home developers in predicting the behavior of the housing market on the short-term. The model utilizes artificial neural networks which are trained using historical market performance data sets in order to predict unforeseen future performance. An application example is analyzed to illustrate the use of the model and demonstrate its capabilities of effectively analyzing and predicting the housing market performance. The model testing and validation showed that the error in prediction is in the range between –2% and +2%.

Final Conclusion:(Research Paper By: Ahmed Khalafallah.):

1.robust in approximating almost any input/output.

2.Several network structures are trained, cross-validated and tested by varying the number of hidden layers, the number of neurons in each hidden layer, the transfer function, the learning method, the cross-validation sample size, and the testing sample size.

REFERENCES:

1.© 2015 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/). Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

2.TSINGHUA SCIENCE AND TECHNOLOGY ISSN 1007-0214 52/67 pp325-328 Volume 13, Number S1, October 2008.

3.Revista de Metodos Cuantitativos para la Economia y la Empresa · June 2013.

4.17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

5.Adana Science and Technology University, Faculty of Engineering and Natural Science, Civil Engineering Department, Adana, Turkey.

6.a Key Laboratory of Land Resources Evaluation and Monitoring in Southwest, Sichuan Normal University, Chengdu 610068, China

a.b Land and Resources Department of Sichuan Province, Chengdu 610072, China.

Higher School of Economics, Faculty of Economics, Sedova St. 55/2, Saint-